



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

## Σύστημα Παρακολούθησης και Ελέγχου Υποσταθμού Με Χρήση Δικτύου LoRaWAN

### ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ Σ. ΑΡΑΠΗ

Επιβλέπων: Παναγιώτης Τσανάκας  
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων: Άρης Ευάγγελος Δημέας  
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2025





ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

## Σύστημα Παρακολούθησης και Ελέγχου Υποσταθμού Με Χρήση Δικτύου LoRaWAN

### ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ Σ. ΑΡΑΠΗ

**Επιβλέπων:** Παναγιώτης Τσανάκας  
Καθηγητής Ε.Μ.Π.

**Συνεπιβλέπων:** Άρης Ευάγγελος Δημέας  
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22 Οκτωβρίου 2025.

(Υπογραφή)

(Υπογραφή)

(Υπογραφή)

.....  
Παναγιώτης Τσανάκας  
Καθηγητής Ε.Μ.Π.

.....  
Γρηγόρης Καραγιώργος  
Επίκουρος Καθηγητής

.....  
Γεώργιος Γεωργίου  
Επιστ. Συνεργάτης

Αθήνα, Οκτώβριος 2025





ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Copyright © - All rights reserved. Με την επιφύλαξη παντός δικαιώματος.  
Θεόδωρος Αράπης, 2025.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέκρινε.

**ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ**

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....  
Θεόδωρος Αράπης

22 Οκτωβρίου 2025



## Περίληψη

---

Η ανάγκη για αποδοτική παρακολούθηση και διαχείριση των ενεργειακών εγκαταστάσεων καθιστά επιτακτική την ανάπτυξη έξυπνων και αυτόνομων συστημάτων παρακολούθησης υποσταθμών. Στην παρούσα διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε ένα ολοκληρωμένο σύστημα απομακρυσμένης παρακολούθησης και ελέγχου ενός ηλεκτρικού υποσταθμού, βασισμένο στην τεχνολογία LoRaWAN.

Το σύστημα περιλαμβάνει δύο τριφασικούς μετρητές, ο καθένας από τους οποίους κατασκευάστηκε με τη χρήση της πλακέτας ανάπτυξης The Things Uno και τρεις αισθητήρες PZEM-004T, ένας ανά φάση, για την καταγραφή μονοφασικών ηλεκτρικών μεγεθών. Η συλλογή των δεδομένων επιτυγχάνεται μέσω ενός LoRaWAN gateway, το οποίο έχει συναρμολογηθεί βασιζόμενο σε ένα Raspberry Pi 4B, μία μονάδα συγκέντρωσης iC880A-SPI και μία κεραία σχετικά χαμηλού κέρδους των 2dBi. Το λογισμικό περιλαμβάνει την εγκατάσταση της στοίβας ανοιχτού κώδικα The Things Stack και του LoRa Basics Station, ενώ τα δεδομένα αποθηκεύονται και προβάλλονται μέσω διαδικτυακής εφαρμογής υλοποιημένης αξιοποιώντας το web framework Spring Boot της JAVA και τη βιβλιοθήκη React της JavaScript.

Σκοπός της εργασίας είναι η πρακτική διερεύνηση της αξιοπιστίας, της επεκτασιμότητας και της χρηστικότητας των LoRaWAN βασισμένων συστημάτων σε κρίσιμες εφαρμογές εποπτείας και αυτοματισμού υποσταθμών ηλεκτρικής ενέργειας.

## Λέξεις Κλειδιά

LoRaWAN, υποσταθμός, απομακρυσμένη παρακολούθηση, Raspberry Pi, The Things Stack, LoRa Basics Station, end device, IoT, Spring Boot, React



## **Abstract**

---

The need for efficient monitoring and management of energy installations necessitates the development of intelligent and autonomous substation monitoring systems. In this diploma thesis, a complete system for the remote monitoring and control of an electrical substation was designed and implemented, based on LoRaWAN technology.

The system includes two "three-phase" meters, each built using the The Things Uno development board and three PZEM-004T sensors, one per phase, for capturing single-phase electrical measurements. Data collection is achieved through a LoRaWAN gateway based on a Raspberry Pi 4B, an iC880A-SPI concentrator module and a high-gain antenna. The software stack includes the installation of the open-source The Things Stack and the LoRa Basics Station, while data is stored and visualized via a web application developed using Java Spring Boot and React.

The purpose of this work is to practically evaluate the reliability, scalability and usability of LoRaWAN-based systems in critical applications related to the supervision and automation of electrical substations.

## **Keywords**

LoRaWAN, substation, remote monitoring, Raspberry Pi, The Things Stack, LoRa Basics Station, end device, IoT, Spring Boot, React



*Στους γονείς μου*



## Ευχαριστίες

---

Θα ήθελα να εκφράσω την εκτίμησή μου προς τον Καθηγητή κ. Παναγιώτη Τσανάκα για την εμπιστοσύνη που επέδειξε με την ανάθεση της παρούσας διπλωματικής εργασίας. Ιδιαίτερη μνεία αξίζει στον Καθηγητή κ. Άρη Δημέα, τον οποίο ευχαριστώ θερμά για την επιστημονική του καθοδήγηση, την ουσιαστική επίβλεψη και την άριστη συνεργασία κατά τη διάρκεια της εκπόνησης αυτής της εργασίας. Τέλος, θα ήθελα να αναγνωρίσω τη συμβολή των γονέων μου, οι οποίοι με στήριξαν έμπρακτα και ηθικά καθ' όλη τη διάρκεια των σπουδών μου.

Αθήνα, Οκτώβριος 2025

Θεόδωρος Αράπης



# Περιεχόμενα

---

<b>Περίληψη</b>	<b>1</b>
<b>Abstract</b>	<b>3</b>
<b>Ευχαριστίες</b>	<b>7</b>
<b>1 Εισαγωγή</b>	<b>21</b>
1.1 Αντικείμενο της διπλωματικής εργασίας . . . . .	22
1.2 Οργάνωση του τόμου . . . . .	23
<b>I Θεωρητικό Μέρος</b>	<b>25</b>
<b>2 Τεχνολογίες LPWAN και το πρωτόκολλο LoRaWAN</b>	<b>27</b>
2.1 Εισαγωγή στα LPWAN . . . . .	27
2.2 Σύγκριση τεχνολογιών LPWAN . . . . .	28
2.3 Τεχνολογία LoRa . . . . .	31
2.3.1 Γενική επισκόπηση της τεχνολογίας LoRa . . . . .	31
2.3.2 Ραδιοφωνική διάδοση σε αστικό περιβάλλον . . . . .	32
2.3.3 Chirp Spread Spectrum και η υλοποίησή του στη LoRa . . . . .	38
2.3.4 Παράγοντας εξάπλωσης και ευαισθησία δέκτη . . . . .	41
2.3.5 Ρυθμός συμβόλων, Chips, Bitrate και εξισώσεις . . . . .	43
2.3.6 Υπολογισμός χρόνου εκπομπής (Time-on-Air) και παράγοντες που τον επηρεάζουν . . . . .	46
2.3.7 Αποδιαμόρφωση και αποκαδικοποίηση σήματος LoRa . . . . .	48
2.4 Το πρωτόκολλο LoRaWAN . . . . .	49
2.4.1 Αρχιτεκτονική δικτύου LoRaWAN . . . . .	50
2.4.2 Κλάσεις συσκευών και χρονισμοί επικοινωνίας . . . . .	53
2.4.2.1 Class A - Υποχρεωτική (βασική) κλάση, ελάχιστης ισχύος . . . . .	53
2.4.2.2 Class B - Προγραμματισμένα παράθυρα λήψης με φάρο συγχρονισμού . . . . .	54
2.4.2.3 Class C - Συνεχής λήψη (χαμηλή λανθάνουσα, υψηλότερη ισχύς) . . . . .	56
2.4.3 Τύποι μηνυμάτων και δομή πλαισίου LoRaWAN . . . . .	57
2.4.4 Ενεργοποίηση συσκευών και Ασφάλεια (OTAA vs ABP) . . . . .	60
2.4.4.1 Over-The-Air Activation (OTAA) . . . . .	61
2.4.4.2 Activation By Personalization (ABP) . . . . .	64

2.4.4.3 Ασφάλεια E2E και κρυπτογραφία . . . . .	65
2.4.5 Ρυθμός μετάδοσης, ADR, χρόνος στον αέρα (ToA) και περιορισμοί εκπομπής . . . . .	66
2.4.5.1 Adaptive Data Rate (ADR) . . . . .	67
2.4.5.2 Περιορισμοί Duty Cycle και κανονισμοί περιοχής . . . . .	68
<b>3 Τεχνολογική στοίβα λογισμικού του συστήματος</b>	<b>71</b>
3.1 The Things Stack . . . . .	71
3.2 LoRa Basics™ Station . . . . .	75
3.3 Docker . . . . .	78
3.4 Spring Boot . . . . .	81
3.5 ReactJS . . . . .	83
3.6 PostgreSQL . . . . .	84
<b>4 Υλικό και σχεδιασμός συσκευών του συστήματος</b>	<b>87</b>
4.1 Γενική αρχιτεκτονική συστήματος . . . . .	87
4.2 LoRaWAN Gateway . . . . .	87
4.2.1 Υλικά και κατασκευή . . . . .	87
4.2.2 Συνδεσμολογία και λειτουργία . . . . .	90
4.3 Τριφασικοί Μετρητές . . . . .	91
4.3.1 Υλικά και αρχιτεκτονική . . . . .	91
4.3.2 Συνδεσμολογία και λειτουργία . . . . .	92
<b>II Πρακτικό Μέρος</b>	<b>97</b>
<b>5 Πρακτική υλοποίηση του συστήματος</b>	<b>99</b>
5.1 Προετοιμασία LoRaWAN gateway . . . . .	99
5.1.1 Διασύνδεση Hardware . . . . .	99
5.1.2 Εγκατάσταση λειτουργικού συστήματος . . . . .	100
5.1.3 Εγκατάσταση Docker και Docker Compose . . . . .	104
5.1.4 Εγκατάσταση του The Things Stack . . . . .	106
5.1.4.1 Λήψη πηγαίου κώδικα . . . . .	106
5.1.4.2 Παραμετροποίηση docker-compose . . . . .	106
5.1.5 Εγκατάσταση του LoRa Basics Station . . . . .	110
5.1.5.1 Λήψη πηγαίου κώδικα . . . . .	110
5.1.5.2 Υπολογισμός Gateway EUI . . . . .	110
5.1.5.3 Καταχώριση του gateway στο The Things Stack . . . . .	110
5.1.5.4 Παραμετροποίηση docker-compose . . . . .	112
5.2 Προετοιμασία τριφασικού μετρητή . . . . .	116
5.2.1 Διασύνδεση Hardware . . . . .	116
5.2.2 Προγραμματισμός του The Things Uno . . . . .	117
5.2.2.1 Εγκατάσταση και ρύθμιση Arduino IDE . . . . .	117
5.2.2.2 Δομή κώδικα και ροή λειτουργίας . . . . .	119

---

5.2.3 Καταχώριση του The Things Uno στο The Things Stack . . . . .	124
5.2.3.1 Δημιουργία Application . . . . .	124
5.2.3.2 Εγγραφή TTU στο Application . . . . .	125
5.2.4 Κωδικοποίηση Payload Formatter στο TTS . . . . .	127
5.3 Πειραματική δοκιμή λειτουργείας . . . . .	130
<b>6 Ανάπτυξη Web Εφαρμογής για την οπτικοποίηση των δεδομένων</b>	<b>135</b>
6.1 Γενική αρχιτεκτονική εφαρμογής . . . . .	135
6.2 Backend: Spring Boot REST API και ασφάλεια . . . . .	136
6.2.1 Δομή πακέτων, βασικών κλάσεων και βάσης δεδομένων . . . . .	136
6.2.2 Διαχείριση εισερχόμενων uplinks μέσω webhook . . . . .	137
6.2.3 REST API προς την React εφαρμογή . . . . .	139
6.3 Frontend: React διεπαφή χρήστη . . . . .	139
6.3.1 Δομή και βασικά components . . . . .	139
6.3.2 Διαχείριση JWT και κλήσεις προς το REST API . . . . .	140
6.3.3 Γραφική αναπαράσταση και επιλογές χρήστη . . . . .	140
6.4 Εκκίνηση εφαρμογής και παρουσίαση λειτουργειών . . . . .	141
6.5 Ολοκλήρωση εγκατάστασης στο Raspberry Pi: ενοποιημένη εκκίνηση υπηρεσιών και auto-start στο boot . . . . .	147
6.5.1 Σενάρια start/stop: εκκίνηση όλων των υπηρεσιών «με ένα κλικ» . . . . .	147
6.5.2 Αυτόματη εκκίνηση στο boot με systemd . . . . .	151
<b>III Επίλογος</b>	<b>153</b>
<b>7 Συμπεράσματα και Μελλοντικές Κατευθύνσεις</b>	<b>155</b>
7.1 Συμπεράσματα . . . . .	155
7.2 Μελλοντικές κατευθύνσεις . . . . .	156
<b>Βιβλιογραφία</b>	<b>163</b>
<b>Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια</b>	<b>165</b>



## Κατάλογος Σχημάτων

---



# Κατάλογος Εικόνων

---

2.1	Σύγκριση τεχνολογιών ασύρματης επικοινωνίας (LPWAN) ως προς τον ρυθμό μετάδοσης, την κατανάλωση ενέργειας, την εμβέλεια και το κόστος. . . . .	28
2.2	Μοντέλο OSI σε αντιστοίχιση με τα LoRa και LoRaWAN επίπεδα. . . . .	31
2.3	Zώνη Fresnel με 40% της κάλυψης από εμπόδια. . . . .	33
2.4	Σύγκριση ευαισθησίας LoRa και FSK, υπογραμμίζοντας την υπεροχή της τεχνολογίας LoRa σε συνθήκες χαμηλού λόγου σήματος προς θόρυβο (SNR). . . . .	36
2.5	Παράλληλη σύγκριση ενός down-chirp (αριστερά) και ενός up-chirp (δεξιά), που δείχνει τη γραμμική μείωση/αύξηση της στιγμιαίας συχνότητας (πάνω) συνοδευόμενη από το αντίστοιχο σήμα στο πεδίο του χρόνου, όπου η απόσταση των κυμάτων εκτείνεται/συμπλέζεται καθ' όλη τη διάρκεια του συμβόλου. . . . .	40
2.6	Φασματογράφημα σήματος LoRa που παρουσιάζει 8 αρχικά up-chirps προ-οιμίου, 2 down-chirps συγχρονισμού και ακολουθία 5 chirp με κωδικοποιημένα δεδομένα (διαφορετική κυκλική μετατόπιση σε κάθε σύμβολο). Το σήμα σαρώνει πλήρως ένα εύρος ζώνης 125kHz με γραμμικά αυξανόμενη ή μειούμενη συχνότητα σε κάθε chirp. . . . .	40
2.7	Σχηματική απεικόνιση της κυματομορφής Chirp Spread Spectrum στο LoRa για Spreading Factor (4 bits). Το διάγραμμα δείχνει πώς κάθε σύμβολο σαρώνει γραμμικά το εύρος ζώνης από $f_{low}$ σε $f_{high}$ , καθώς και την αντιστοίχιση των chips σε διακριτές τιμές (0-15) μέσα σε ένα σύμβολο. . . . .	41
2.8	Απεικόνιση κυματομορφών Chirp Spread Spectrum στο LoRa για Spreading Factor 4, με ένδειξη των διακριτών τιμών (0, 4, 8, 12) στην αντίστοιχη χρονική θέση μέσα στο σύμβολο $T_s$ . Τα μπλε σημάδια δείχνουν το κλάσμα της περιόδου συμβόλου στο οποίο εκπέμπεται κάθε τιμή, ενώ η πράσινη σήμανση αντιστοιχεί στον αριθμό του chip. . . . .	44
2.9	Τεχνολογική στοίβα των LoRa και LoRaWAN. . . . .	50
2.10	Τυπική αρχιτεκτονική LoRaWAN δικτύου. . . . .	51
2.11	Αρχιτεκτονική LoRaWAN Network Server. . . . .	52
2.12	Ροή Uplink και Downlink μηνυμάτων στο Πρωτόκολλο LoRaWAN. . . . .	54
2.13	Ροή επικοινωνίας κλάσης A. . . . .	55
2.14	Ροή επικοινωνίας κλάσης B. . . . .	55
2.15	Ροή επικοινωνίας κλάσης C. . . . .	56
2.16	Δομή πακέτου LoRaWAN. . . . .	57
2.17	Ροή μηνυμάτων για ενεργοποίηση OTAA στο LoRaWAN 1.0. . . . .	62
2.18	Ροή μηνυμάτων για ενεργοποίηση OTAA στο LoRaWAN 1.1. . . . .	63

2.19 Προ-διαμοιρασμός του DevAddr και των session keys για ενεργοποίηση ABP στο LoRaWAN 1.0. . . . .	65
2.20 Προ-διαμοιρασμός του DevAddr και των session keys για ενεργοποίηση ABP στο LoRaWAN 1.1. . . . .	65
3.1 Αρχιτεκτονική του The Things Stack με τα βασικά του δομικά στοιχεία. . . . .	72
3.2 Αρχιτεκτονική του LoRa Basics <sup>TM</sup> Station. . . . .	76
3.3 Αρχιτεκτονική Docker: ο Docker Client εκτελεί εντολές (build, pull, run) προς τον Docker daemon, που δημιουργεί/χειρίζεται images και containers και επικοινωνεί με τον Docker Registry. . . . .	80
3.4 Αρχιτεκτονική ενός Web Application, αποτελούμενο από ένα React Frontend App που επικοινωνεί μέσω Axios/REST APIs με ένα Spring Boot Backend App, μετατρέπει δεδομένα DTOs (Data Transfer Objects) σε οντότητες (Entities) και εκτελεί διεργασίες αποθήκευσης/ανάκτησης αυτών των δεδομένων από μία PostgreSQL βάση δεδομένων. . . . .	84
4.1 Τοπολογία του συστήματος. . . . .	87
4.2 Υλοποίηση LoRaWAN Gateway και σύνδεσμολογία εξαρτημάτων. . . . .	90
4.3 Υλοποίηση τριφασικού μετρητή - σύνδεση The Things Uno με 3 PZEM-004T V3.0 . . . . .	93
5.1 Τελική συνδεσμολογία των εξαρτημάτων του LoRaWAN gateway. . . . .	99
5.2 Raspberry Pi Imager. . . . .	100
5.3 Apply OS Customisation settings Option. . . . .	100
5.4 General OS Customisation settings. . . . .	101
5.5 Services OS Customisation settings. . . . .	102
5.6 Εργαλείο παραμετροποίησης λογισμικού του Raspberry Pi . . . . .	103
5.7 Επιλογή ενεργοποίησης διεπαφής SPI . . . . .	103
5.8 Επιλογή καταχώρησης νέου Gateway. . . . .	111
5.9 Εισαγωγή EUI. . . . .	111
5.10 Δημιουργία Gateway ID και name, επιλογή Frequency plan και δημιουργία κλειδιού LNS. . . . .	111
5.11 Αποθήκευση API κλειδιού. . . . .	112
5.12 Gateway Overview and status στο TTS Console. . . . .	116
5.13 Τελική συνδεσμολογία των εξαρτημάτων του τριφασικού μετρητή. . . . .	116
5.14 Λογισμικό Arduino IDE. . . . .	117
5.15 Αναζήτηση και επιλογή πλακέτας Arduino Leonardo. . . . .	117
5.16 Επιλογή Arduino Leonardo στη θύρα COM3 όπου έχει συνδεθεί το The Things Uno. . . . .	118
5.17 Κάνουμε upload to sketch στο TTU. . . . .	118
5.18 Εξόδος του sketch Deviceinfo στο Serial Monitor με τις πληροφορίες της συσκευής. . . . .	119
5.19 Επιλογή προσθήκης νέου Application. . . . .	125
5.20 Ορισμός Application ID και Application name. . . . .	125

5.21	Επιλογή εγγραφής end device. . . . .	125
5.22	Επιλογή The Things Uno μοντέλου από Device registry του The Things Stack.	126
5.23	Συμπλήρωση Provisioning information για το νέο end device. . . . .	127
5.24	Η καταχωρημένη τερματική συσκευή στο μενού της εφαρμογής. . . . .	127
5.25	Κωδικοποίηση του Default uplink payload formatter. . . . .	128
5.26	Δοκιμαστικό κύκλωμα τριφασικού μετρητή με τρεις λαμπτήρες ως φορτία (φάσεις L1-L3). Το καλώδιο φάσης του ρευματολήπτη (μάυρο) χωρίζεται σε 3 καλώδια (κόκκινο, καφέ, μαύρο), τα οποία συνδέονται με ένα PZEM και ένα λαμπτήρα, αντιπροσωπεύοντας μία «υπο-φάση» το καθένα. Το καλώδιο ουδετέρου του ρευματολήπτη (μπλε) συνδέεται σε όλα τα PZEM και τους λαμπτήρες.	130
5.27	Ροή εντολών MAC και επιτυχές OTAA join της συσκευής στο EU868 (ρυθμίσεις καναλιών/ρυθμών, DevAddr, DR5, αποτυχημένη αλλαγή κλάσης ως αναμενόμενο).	131
5.28	Ένδειξη Serial Monitor με μετρήσεις από 3 PZEM-004T και αποστολή uplink (mac tx uncnf) στο TTN. . . . .	132
5.29	Πίνακας Live data στο TTS: χρονολόγιο γεγονότων για join-accept και εισερχόμενα uplinks. Διακρίνονται τα πεδία DevAddr, η επισήμανση Payload με δεκαεξαδικά δεδομένα, καθώς και η προεπισκόπηση decoded_payload όταν είναι ενεργός ο formatter. . . . .	133
5.30	decoded_payload από τον formatter: ανά sensor προβάλλονται voltage, current, power, energy, frequency, powerFactor. Οι τιμές προκύπτουν από τις κλίμακες κωδικοποίησης του uplink και είναι έτοιμες για αποθήκευση/οπτικοποίηση χωρίς περαιτέρω μετασχηματισμούς. . . . .	134
6.1	Στην εφαρμογή 3-Phase Power Meters, φαίνεται το ενεργό webhook με ID power-monitoring-app-webhook και Base URL <a href="http://192.168.0.100:8080/lorawan-data">http://192.168.0.100:8080/lorawan-data</a> . . . . .	138
6.2	Σελίδα Login της διαδικτυακής εφαρμογής. . . . .	142
6.3	Επιλογή μετρητή προς απεικόνιση. . . . .	142
6.4	Επιλογή μετρητή προς απεικόνιση. . . . .	143
6.5	Επιλογή μορφής διαγράμματος. . . . .	143
6.6	Επιλογή κάτω ορίου του χρονικού διαστήματος των δεδομένων. . . . .	143
6.7	Βασική σελίδα (Dashboard) της διαδικτυακής εφαρμογής. . . . .	144
6.8	Πίνακας με την αναλυτική παρουσίαση των δεδομένων. . . . .	144
6.9	Line Chart. . . . .	145
6.10	Bar Chart. . . . .	145
6.11	Pie Chart (Average by phase). . . . .	146
6.12	Pie Chart. . . . .	146



# Κατάλογος Πινάκων

---

2.1	Συγκριτικός πίνακας τεχνολογιών NB-IoT, LTE-M και LoRa. . . . .	29
2.2	Πίνακας παραμέτρων Spreading Factor και αντίστοιχων ορίων SNR. . . . .	42
2.3	'Ορια SNR και ευαισθησία δέκτη για διαφορετικά Spreading Factors, με $BW = 125\text{kHz}$ και $NF = 6\text{dB}$ . . . . .	43
2.4	Διάρκεια συμβόλου LoRa σε $BW=125\text{ kHz}$ για διάφορα Spreading Factors, με ενεργοποιημένη Low Data Rate Optimization (LDRO) όπου απαιτείται. . .	48
2.5	Συνοπτικός πίνακας βασικών MAC εντολών LoRaWAN. . . . .	60
2.6	Πεδία Join-Request στο LoRaWAN 1.0. . . . .	61
2.7	Πεδία Join-Request στο LoRaWAN 1.1. . . . .	61
2.8	Πεδία Join-Accept στο LoRaWAN 1.0. . . . .	62
2.9	Πεδία Join-Accept στο LoRaWAN 1.1. . . . .	62
2.10	EU863-870 (EU868): μέγιστα μεγέθη MACPayload και ωφέλιμου App Payload, repeater-compatible. Το App Payload υπολογίζεται από MACPayload με αφαίρεση του LoRaWAN overhead (Θεωρώντας άδειο FOpts). . . . .	68
4.1	Βασικά υλικά για το LoRaWAN Gateway. . . . .	88
4.2	Τεχνικά χαρακτηριστικά Raspberry Pi 4 Model B. . . . .	89
4.3	Τεχνικά χαρακτηριστικά του iC880A-SPI LoRaWAN Concentrator. . . . .	89
4.4	Συνδεσμολογία ακίδων (pins) iC880A-SPI με Raspberry Pi 4 Model B. . . .	91
4.5	Κατάλογος υλικών για έναν τριφασικό μετρητή . . . . .	91
4.6	Τεχνικά χαρακτηριστικά PZEM-004T V3.0 . . . . .	92
4.7	Τεχνικά χαρακτηριστικά του The Things Uno . . . . .	92
4.8	Συνδεσμολογία 1 <sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 1 . . . .	93
4.9	Συνδεσμολογία 2 <sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 2 . . . .	94
4.10	Συνδεσμολογία 3 <sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 3 . . . .	94
5.1	Διάταξη payload ανά φάση (14 bytes/phase, σύνολο 42 bytes για 3 PZEM). .	124
6.1	Πεδία του πίνακα meter. . . . .	137
6.2	Πεδία του πίνακα sensor_data. . . . .	137
6.3	Πεδία του πίνακα users. . . . .	137



## Κεφάλαιο 1

### Εισαγωγή

---

Σε μια εποχή σημαδεμένη από διαρκείς τεχνολογικές εξελίξεις, η ταχύτατη ανάπτυξη του Διαδικτύου των Πραγμάτων Internet of Things - IoT ήταν αναμενόμενη, γεγονός που έχει φέρει στο προσκήνιο πλήθος από νέες τεχνολογίες επικοινωνίας, ικανές να συνδέουν μία τεράστια ποικιλία απομακρυσμένων αισθητήρων και συσκευών, με μεγάλο εύρος λειτουργίας και χαμηλό ενεργειακό κόστος [1]. Μάλιστα, εκτιμάται ότι μέχρι το 2030 θα υπάρχουν περισσότερες από 30 δισεκατομμύρια συσκευές συνδεδεμένες στο διαδίκτυο παγκοσμίως [2]. Η έννοια του Διαδικτύου των Πραγμάτων αναφέρεται σε ένα δίκτυο επικοινωνίας, το οποίο συγκροτείται από πληθώρα συσκευών εξοπλισμένων με αισθητήρες. Οι συσκευές αυτές συλλέγουν δεδομένα από το φυσικό περιβάλλον, τα μεταδίδουν, τα διαμοιράζονται και τα αξιοποιούν, με σκοπό την παροχή ποικίλων υπηρεσιών.

Μία από τις πλέον πολλά υποσχόμενες τεχνολογίες στον χώρο των ασύρματων δικτύων ευρείας περιοχής χαμηλής ισχύος (Low Power Wide Area Networks - LPWANs) είναι το LoRaWAN. Η τεχνολογία αυτή επιτρέπει τη δημιουργία ανθεκτικών και κλιμακούμενων υποδομών επικοινωνίας με ελάχιστες ενεργειακές απαιτήσεις, καθιστώντας την ιδανική για εφαρμογές όπου η συνδεσιμότητα και η αυτονομία είναι υψηλή σημασίας.

Στον τομέα της ενέργειας και ειδικότερα στην παρακολούθηση και τον έλεγχο υποσταθμών μέσης και χαμηλής τάσης, η ανάγκη για απομακρυσμένη συλλογή μετρήσεων και δεδομένων, καθώς και του εξ αποσάσεως ελέγχου, είναι πιο επίκαιρη από ποτέ. Οι «έξυπνοι» μετρητές και τα συστήματα τηλεμετρίας επιτρέπουν την πρόβλεψη, την αποδοτικότερη κατανομή και απαραίτητη εποπτεία της κατανάλωσης, τη βελτίωση της ποιότητας της ενέργειας και την έγκαιρη ανίχνευση σφαλμάτων. Συνεπώς, η ενσωμάτωση αυτών των δυνατοτήτων με τεχνολογίες όπως το LoRaWAN αποτελεί ένα σημαντικό βήμα προς την υλοποίηση και εφαρμογή των «έξυπνων» δικτύων ενέργειας (smart grids).

Η παρούσα διπλωματική εργασία αποσκοπεί στη μελέτη, σχεδίαση και υλοποίηση ενός ολοκληρωμένου συστήματος παρακολούθησης και ελέγχου ηλεκτρικού υποσταθμού, με τη χρήση του LoRaWAN. Το σύστημα που αναπτύχθηκε περιλαμβάνει:

- δύο τριφασικούς μετρητές, βασισμένους στην πλακέτα ανάπτυξης The Things Uno της The Things Industries και σε τρεις (ανά μετρητή) μονοφασικούς αισθητήρες PZEM-004T για τη μέτρηση ηλεκτρικών μεγεθών,
- έναν LoRaWAN gateway, κατασκευασμένο με ένα Raspberry Pi 4B και μία μονάδα συγκέντρωσης iC880A-SPI, στην οποία συνδέεται μία κεραία χαμηλού σχετικά κέρδους

των 2dBi,

- ανάπτυξη δυναμικής ιστοσελίδας με χρήση Java Spring Boot και React (TypeScript), με σκοπό την παρουσίαση των μετρήσεων που λαμβάνονται από τους μετρητές.

Η υλοποίηση αυτή δεν περιορίζεται μόνο στη θεωρητική διερεύνηση του προαναφερόμενου συστήματος, αλλά αποσκοπεί, επιπλέον, στην πρακτική αξιολόγηση της τεχνολογίας LoRaWAN στο πεδίο ενός IoT συστήματος παρακολούθησης υποσταθμού, επιδιώκοντας να αναδείξει τις δυνατότητες και τους περιορισμούς της, όταν εφαρμόζεται σε ένα πραγματικό περιβάλλον ενεργειακής υποδομής.

## 1.1 Αντικείμενο της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη, σχεδίαση και υλοποίηση ενός ολοκληρωμένου, χαμηλού κόστους και ενεργειακά αποδοτικού συστήματος απομακρυσμένης παρακολούθησης και ελέγχου ενός ηλεκτρικού υποσταθμού, αξιοποιώντας τις δυνατότητες του πρωτοκόλλου LoRaWAN. Το σύστημα αυτό εντάσσεται στο πλαίσιο των έξυπνων ενεργειακών υποδομών και της τεχνολογίας Internet of Things (IoT) και έχει στόχο τη δημιουργία μιας επεκτάσιμης και παράλληλα αξιόπιστης αρχιτεκτονικής, η οποία θα μπορεί να υιοθετείται και σε άλλα βιομηχανικά ή ενεργειακά περιβάλλοντα.

Ο στόχος της εργασίας είναι διπτός:

1. **Πρακτική αξιοποίηση του LoRaWAN** ως βασικού μέσου επικοινωνίας για την αποστολή δεδομένων από απομακρυσμένες συσκευές μέτρησης προς ένα κεντρικό σύστημα διαχείρισης. Η τεχνολογία αυτή προσφέρει σημαντικά πλεονεκτήματα όπως μεγάλη εμβέλεια, χαμηλή κατανάλωση ενέργειας και υποστήριξη για μεγάλο αριθμό συσκευών, καθιστώντας την, επομένως, ιδανική για απαιτητικά περιβάλλοντα όπως οι ηλεκτρικοί υποσταθμοί.
2. **Ανάπτυξη ολοκληρωμένης πλατφόρμας εποπτείας και διαχείρισης ενεργειακών μετρήσεων**, η οποία επιτρέπει την οπτικοποίηση και αποθήκευση κρίσιμων ηλεκτρικών παραμέτρων σε πραγματικό χρόνο, όπως η τάση, το ρεύμα, η ισχύς και η κατανάλωση ενέργειας, ανά φάση. Η πλατφόρμα προσφέρει δυνατότητα επεκτασιμότητας και προσαρμοστικότητας, προκειμένου να μπορεί να υποστηρίξει πολλαπλά σημεία μέτρησης, καθώς και μελλοντική ενσωμάτωση επιπρόσθετων λειτουργιών, όπως ειδοποιήσεις ή αυτοματισμούς.

Η υλοποίηση βασίστηκε σε πλήρως παραμετροποιήσιμα εξαρτήματα ανοιχτού κώδικα και λογισμικού, όπως οι συσκευές The Things Uno, οι αισθητήρες PZEM-004T, ο συγκεντρωτής σήματος iC880A-SPI, το Raspberry Pi 4B, και η πλατφόρμα The Things Stack. Τα δεδομένα διαχειρίζονται και παρουσιάζονται μέσω διαδικτυακής εφαρμογής υλοποιημένης σε εργαλέια ανοιχτού κώδικα, όπως το JAVA Spring Boot framework και η front-end βιβλιοθήκη React της JavaScript.

Τέλος, η εργασία αυτή συμβάλλει στην αξιολόγηση των τεχνικών δυνατοτήτων του LoRaWAN, όταν αυτό χρησιμοποιείται σε κρίσιμες εφαρμογές παρακολούθησης, εντοπισμού

ανωμαλιών και ενεργειακής διαχείρισης, προσφέροντας έτσι ένα αξιόπιστο πρότυπο, το οποίο να μπορεί να αναπαραχθεί ή να εξελιχθεί περαιτέρω σε μεγαλύτερα δίκτυα ή άλλους τύπους βιομηχανικών εγκαταστάσεων.

## 1.2 Οργάνωση του τόμου

Η εργασία είναι οργανωμένη ως εξής:

- **Κεφάλαιο 2** - Παρουσιάζονται τα θεμέλια των LPWAN, η φυσική τεχνολογία LoRa (διάδοση, CSS, SF/BW, ρυθμοί) και το πρωτόκολλο LoRaWAN (πλαίσια, MAC εντολές, ενεργοποίηση, περιορισμοί ζωνών όπως EU868).
- **Κεφάλαιο 3** - Περιγράφεται η τεχνολογική στοίβα του συστήματος: The Things Stack ως LNS, LoRa Basics Station ως packet forwarder, υποδομή Docker/Compose, PostgreSQL, καθώς και βοηθητικές υπηρεσίες/εργαλεία.
- **Κεφάλαιο 4** - Αναλύεται ο υλικός εξοπλισμός και ο σχεδιασμός: Raspberry Pi 4B, συγκεντρωτής iC880A-SPI, συνδεσμολογία SPI/GPIO, τριφασικοί μετρητές με PZEM-004T v3.0 και The Things Uno.
- **Κεφάλαιο 5** - Παρουσιάζεται η πρακτική υλοποίηση: προετοιμασία gateway, εγκατάσταση/παραμετροποίηση TTS και Basics Station, κώδικας uplink και πειραματικό κύκλωμα.
- **Κεφάλαιο 6** - Περιγράφεται η web εφαρμογή (backend με REST API και frontend dashboard), οι επιλογές χρήστη, και παρουσιάζονται μετρήσεις/διαγράμματα που αξιολογούν τη λειτουργία του συστήματος. Περιλαμβάνονται οδηγίες εκκίνησης/αυτοματοποίησης του συνολικού συστήματος (systemd).
- **Κεφάλαιο 7** - Συνοψίζονται τα συμπεράσματα και προτείνονται σαφείς μελλοντικές κατευθύνσεις για κλιμάκωση, ασφάλεια, αναλυτική επεξεργασία και επιχειρησιακή ωρίμανση.



## Μέρος I

### Θεωρητικό Μέρος



## Κεφάλαιο 2

# Τεχνολογίες LPWAN και το πρωτόκολλο LoRaWAN

---

### 2.1 Εισαγωγή στα LPWAN

Η διαρκώς αυξανόμενη ανάγκη για απομακρυσμένη και ταυτόχρονα αποδοτική, ως προς την ενέργεια, επικοινωνία μεταξύ έξυπνων συσκευών και αισθητήρων έχει οδηγήσει στην εμφάνιση και εξέλιξη μιας νέας γενιάς ασύρματων τεχνολογιών, γνωστών ως Low Power Wide Area Networks (LPWAN). Οι τεχνολογίες LPWAN επιτρέπουν την αποστολή μικρών σε ποσότητα δεδομένων σε μεγάλες αποστάσεις με εξαιρετικά χαμηλή κατανάλωση ενέργειας, καθιστώντας τις ιδιαίτερες για εφαρμογές Internet of Things (IoT), όπου η διάρκεια ζωής της μπαταρίας και η αξιοπιστία είναι κρίσιμοι παράγοντες.

Σε αντίθεση με τις τεχνολογίες Wi-Fi ή Bluetooth, οι οποίες είναι σχεδιασμένες για υψηλούς ρυθμούς μετάδοσης δεδομένων σε μικρές αποστάσεις, τα LPWAN είναι προσανατολισμένα στην υποστήριξη ενός μεγάλου αριθμού συσκευών, με δυνατότητα μετάδοσης δεδομένων σε αποστάσεις που υπερβαίνουν τα 10 χιλιόμετρα, σε ανοικτό πεδίο και σε συχνότητες που βρίσκονται στο μη αδειοδοτημένο φάσμα (unlicensed spectrum). Το σημαντικότερο, μάλιστα, όφελος έναντι άλλων τεχνολογιών μετάδοσης πληροφορίας μεγάλου εύρους (όπως το φάσμα κινητής τηλεφωνίας 3G, 4G ή 5G), είναι η ελάχιστη ενέργεια που απαιτείται για την τροφοδοσία των αντίστοιχων συσκευών [3].

Οι πιο διαδεδομένες τεχνολογίες LPWAN είναι οι εξής:

- **NB-IoT (Narrowband Internet of Things):** αποτελεί τεχνολογία ασύρματης επικοινωνίας και χαμηλής ισχύος, βασισμένη στο LTE (Long-Term Evolution), η οποία λειτουργεί στο αδειοδοτημένο φάσμα και προσφέρει αξιόπιστη κάλυψη εντός κτιρίων (deep indoor penetration). Αναπτύχθηκε από το 3<sup>rd</sup> Generation Partnership Project (3GPP) και υποστηρίζεται από το πρότυπο 3GPP Release 13. Έχει σχεδιαστεί για εφαρμογές με ανάγκες μαζικής συνδεσιμότητας και μικρού όγκου δεδομένων, όπως μετρητές νερού ή αερίου. Η χαμηλή κατανάλωση ενέργειας που απαιτείται για τη λειτουργία των συσκευών έχει ως αποτέλεσμα η διάρκεια λειτουργίας τους να φτάνει έως και 10 χρόνια, με τη χρήση μίας μόνο μπαταρίας [4].
- **LTE-M (LTE Cat-M1):** επίσης βασίζεται στο LTE και προσφέρει υψηλότερους ρυθμούς μετάδοσης δεδομένων από το NB-IoT (έως και 1Mbps), διατηρώντας ωστόσο εξίσου χα-

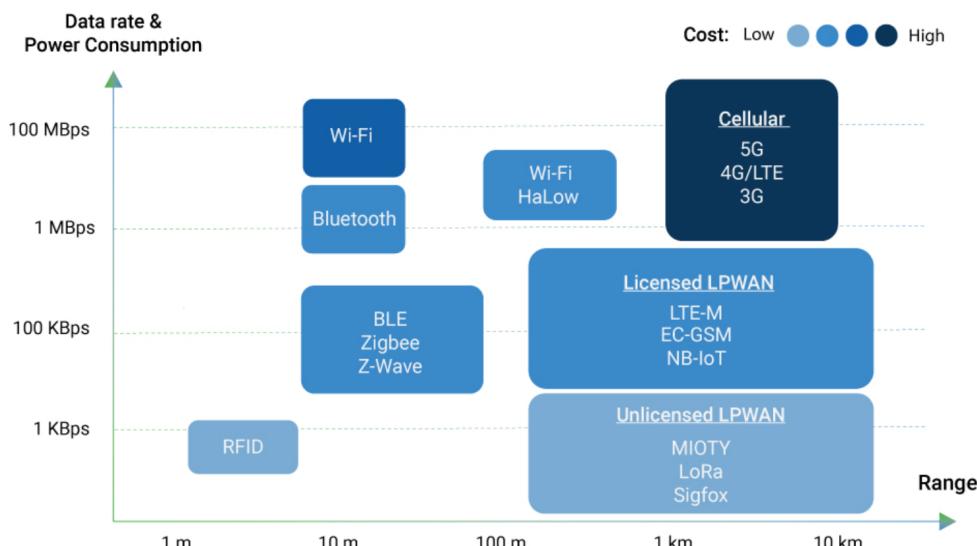
μηλή κατανάλωση [5]. Είναι κατάλληλο για φορητές εφαρμογές που απαιτούν αμφίδρομη επικοινωνία σε πραγματικό χρόνο, όπως η παρακολούθηση οχημάτων, οι φορητές ιατρικές συσκευές και οι φορητοί αισθητήρες. Ένα από τα κύρια πλεονεκτήματά του είναι η υποστήριξη κινητικότητας, επιτρέποντας την απρόσκοπη μετάβαση μεταξύ κυψελών, καθώς και η δυνατότητα φωνητικής επικοινωνίας μέσω VoLTE [6].

- **LoRa και LoRaWAN:** πρόκειται για το πιο διαδεδομένο πρωτόκολλο σε μη αδειοδοτημένο φάσμα (π.χ. 868MHz στην Ευρώπη), με κύρια πλεονεκτήματα την ευκολία υλοποίησης, τη μεγάλη αυτονομία (έως και 10 έτη), τη χαμηλή κατανάλωση ισχύος και την υψηλή ευελιξία ανάπτυξης μέσω ιδιωτικών ή δημόσιων δικτύων. Η τεχνολογία LoRa αναπτύχθηκε αρχικά από τη γαλλική Cycleo και κατοχυρώθηκε από τη Semtech, ενώ το LoRaWAN αναπτύσσεται και προτυποποιείται από τη LoRa Alliance [7].

Τα δίκτυα LPWAN ενσωματώνονται όλο και περισσότερο σε κρίσιμες υποδομές, όπως είναι τα συστήματα παρακολούθησης ενέργειας, γεωργίας ακριβείας<sup>1</sup> (precision farming), έξυπνων πόλεων και βιομηχανικής αυτοματοποίησης, προσφέροντας λύσεις υψηλής κάλυψης, ανθεκτικότητας και χαμηλού κόστους εγκατάστασης.

## 2.2 Σύγκριση τεχνολογιών LPWAN

Στον κάτωθι πίνακα παρατίθενται συγκριτικά οι διαφορές τεχνολογιών LPWAN:



Εικόνα 2.1: Σύγκριση τεχνολογιών ασύρματης επικοινωνίας (LPWAN) ως προς τον ρυθμό μετάδοσης, την κατανάλωση ενέργειας, την εμβέλεια και το κόστος.

[9]

Οι τεχνολογίες LPWAN αποτελούν βασικό πυλώνα για την υλοποίηση ενεργειακά αποδοτικών και μεγάλης εμβέλειας εφαρμογών IoT, με διαφορετικές προσεγγίσεις ως προς το

<sup>1</sup>Η γεωργία ακριβείας (precision agriculture) είναι ένα σύγχρονο μοντέλο καλλιέργειας που αξιοποιεί τεχνολογίες πληροφορικής, αισθητήρες, δορυφορικά δεδομένα και αυτοματισμούς, ώστε να βελτιστοποιήσει τη χρήση πόρων (νερό, λιπάσματα, φυτοφάρμακα, ενέργεια) [8]

φάσμα λειτουργίας, την κατανάλωση ισχύος, την κινητικότητα και τη δυνατότητα υποστήριξης ποικίλων τύπων δεδομένων. Ακολούθως παρουσιάζονται τα προαναφερθέντα χαρακτηριστικά για τις τρεις πιο διαδεδομένες τεχνολογίες LPWAN:

Παράμετρος	NB-IoT	LTE-M	LoRa
<b>Τυποποίηση</b>	3GPP	3GPP	LoRa Alliance
<b>Διαμόρφωση</b>	QPSK, 16QAM	QPSK, 16QAM	CSS (Chirp Spread Spectrum)
<b>Φάσμα Συχνοτήτων</b>	Licensed 3GPP (180kHz)	Licensed 3GPP (1.4MHz)	Unlicensed ISM (EU 868MHz)
<b>Κάλυψη (Link Budget)</b>	151dB	146dB	154dB
<b>Μέγιστο Φορτίο</b>	1600 bytes	1000 bytes	242 bytes
<b>Διάρκεια Ζωής Μπαταρίας</b>	έως 10 έτη	έως 2 έτη	έως 10 έτη
<b>Ταχύτητα Μετάδοσης</b>	200kbps	1Mbps	50kbps
<b>Αμφίδρομη Επικοινωνία</b>	Ναι	Ναι	Ναι
<b>Ασφάλεια</b>	3GPP (128-256 bit)	3GPP (128-256 bit)	AES (128 bit)
<b>Κινητικότητα</b>	< 100km/h	< 300km/h	Ναι
<b>QoS</b>	Ναι	Ναι	Όχι

Πίνακας 2.1: Συγκριτικός πίνακας τεχνολογιών NB-IoT, LTE-M και LoRa.

[3, 10, 5, 7]

**Σημείωση για τους αριθμούς:** Οι ρυθμοί και οι επιδόσεις είναι ενδεικτικοί, καθώς επηρεάζονται από εύρος ζώνης, coding rate, επαναλήψεις, περιβάλλον καναλιού και ρυθμίσεις συστημάτων. Για LoRa (EU,  $BW = 125\text{kHz}$ ) το φυσικό bitrate είναι της τάξης των  $\sim 0.3 - 5.5\text{kbps}$  (ανά SF), ενώ με  $BW = 500\text{kHz}$  μπορεί να φτάσει περίπου τα  $22\text{kbps}$  [11, 12]. Για NB-IoT/LTE-M οι τιμές διαφέρουν ανά release/φορέα/ρύθμιση και δίνονται συνήθως ως εύρος.

**Σημείωση για duty cycle (Ευρώπη):** Στο EU863-870 το επιτρεπόμενο duty cycle δεν είναι παντού 1%, αλλά ξαρτάται από υπο-ζώνη (π.χ. 0.1%, 1% ή 10%). Επομένως, ο επιτρεπός ρυθμός αποστολών συνδέεται άμεσα με τον χρόνο στον αέρα (ToA) του κάθε πακέτου.

Η ανάλυση των επιμέρους χαρακτηριστικών των τριών τεχνολογιών δείχνει πως κάθε μία εξυπηρετεί διαφορετικές ανάγκες, ανάλογα με το σενάριο χρήσης και τις απαιτήσεις της εκάστοτε εφαρμογής.

Ξεκινώντας από την κινητικότητα, το LTE-M υπερέχει με διαφορά, καθώς υποστηρίζει μετακινήσεις με ταχύτητες έως και  $300km/h$  και δυνατότητα handover μεταξύ κυψελών, κάτι που καθιστά εφικτή την αξιόπιστη σύνδεση σε περιπτώσεις όπως είναι η παρακολούθηση οχημάτων ή drones εν κινήσει. Από την άλλη μεριά, το NB-IoT παρέχει περιορισμένη κινητικότητα και είναι περισσότερο κατάλληλο για στατικές συσκευές, ενώ το LoRa μπορεί να χρησιμοποιηθεί για κινητές εφαρμογές μόνο αν βρίσκεται εντός εμβέλειας ενός διαθέσιμου gateway, γεγονός που περιορίζει τη χρήση του σε δυναμικά περιβάλλοντα.

Στο πεδίο της μετάδοσης δεδομένων, το LTE-M προσφέρει τους υψηλότερους ρυθμούς ( $1Mbps$ ), καθώς και υποστήριξη φωνητικής επικοινωνίας μέσω VoLTE, χαρακτηριστικά που αποστιάζουν από τις άλλες δύο τεχνολογίες. Αντίθετα, το LoRa περιορίζεται σε πολύ χαμηλούς ρυθμούς (δεν υπερβαίνουν τα  $50kbps$ ) και είναι σχεδιασμένο κυρίως για απλές, οποραδικές μεταδόσεις.

Όσον αφορά το φάσμα λειτουργίας, τόσο το NB-IoT όσο και το LTE-M αξιοποιούν το αδειοδοτημένο φάσμα, γεγονός που προσφέρει πιο σταθερή σύνδεση, μικρότερο λανθάνοντα χρόνο και καλύτερη ποιότητα υπηρεσίας (QoS). Αυτά τα χαρακτηριστικά είναι κρίσιμα για εφαρμογές όπως POS terminals, όπου απαιτείται γρήγορη και αξιόπιστη μετάδοση συναλλαγών. Από την άλλη, το LoRa λειτουργεί σε μη αδειοδοτημένο φάσμα, που αν και μειώνει το κόστος, υπόκειται σε περιορισμούς όπως το duty cycle και το fair access policy, μειώνοντας, έτσι, την αξιοπιστία σε περιβάλλοντα όπου υπάρχει υψηλή κίνηση δεδομένων.

Σε όρους ενεργειακής απόδοσης, το LoRa και το NB-IoT είναι εμφανώς πιο αποτελεσματικά, υποστηρίζοντας διάρκεια μπαταρίας έως και 10 έτη. Το LTE-M, λόγω της μεγαλύτερης κατανάλωσης ισχύος, τείνει να έχει μικρότερη διάρκεια ζωής, συνήθως μεταξύ 1-2 ετών, κάτι που πρέπει να ληφθεί υπόψη σε εφαρμογές όπου η συντήρηση των κόμβων δεν είναι εύκολη.

Ως προς την εμπορική απήχηση, οι τεχνολογίες του 3GPP (NB-IoT και LTE-M) πρωθυόνται κυρίως μέσω παρόχων κινητής τηλεφωνίας και ενσωματώνονται σε λύσεις ευρείας κλίμακας από τη βιομηχανία [13]. Αντίθετα, το LoRaWAN, μέσω της LoRa Alliance, διατίθεται ευρύτερα για αποκεντρωμένες και ιδιωτικές αναπτύξεις, γεγονός που το έχει καταστήσει ιδιαίτερα δημοφιλές σε αγροτικές εφαρμογές, αισθητήρες έξυπνων κτιρίων και περιβαλλοντική παρακολούθηση [14].

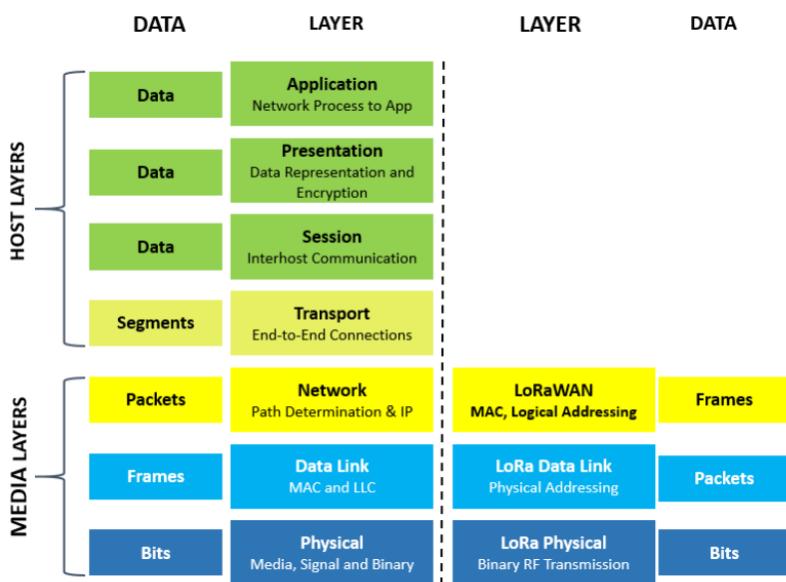
Συνοψίζοντας, δεν υπάρχει μία «καλύτερη» τεχνολογία για κάθε περίπτωση. Η επιλογή εξαρτάται από το εκάστοτε έργο και τους στόχους του: αν προέχει η κινητικότητα και η χαμηλή καθυστέρηση, το LTE-M είναι πιο κατάλληλο, ανν ζητούμενο είναι η μεγάλη διάρκεια ζωής και το χαμηλό κόστος, το LoRa αποτελεί ιδανική επιλογή. Τέλος, το NB-IoT είναι ενδιάμεση λύση για στατικές εφαρμογές με αξιόπιστο σήμα και μεγάλη πυκνότητα κόμβων. Η τελική απόφαση λαμβάνει υπόψη τεχνικούς περιορισμούς, απαιτήσεις απόδοσης και το οικονομικό κόστος υλοποίησης.

## 2.3 Τεχνολογία LoRa

### 2.3.1 Γενική επισκόπηση της τεχνολογίας LoRa

Ξεκινώντας με μία μικρή ιστορική αναδρομή, η τεχνολογία LoRa (Long Range) αναπτύχθηκε αρχικά το 2009 από δύο φίλους, τους Nicolas Sornin και Olivier Seller, όπου στη συνέχεια προστέθηκε στην ομάδα και ένας τρίτος συνεργάτης, ο François Sforza και όλοι μαζί δημιούργησαν τη γαλλική εταιρεία Cycleo το 2010. Δύο χρόνια μετά (2012), η Cycleo εξαγοράστηκε από την αμερικανική εταιρεία Semtech [15]. Η τεχνολογία αυτή λειτουργεί αποκλειστικά στο φυσικό επίπεδο (Physical Layer, PHY) του μοντέλου αναφοράς OSI (Open Systems Interconnection model) και βασίζεται στη διαμόρφωση εξάπλωσης φάσματος τύπου Chirp Spread Spectrum (CSS), που επιτρέπει την αξιόπιστη και χαμηλής κατανάλωσης μετάδοση δεδομένων σε μεγάλες αποστάσεις. Χρησιμοποιεί το ελεύθερο φάσμα ραδιοσυχνοτήτων ISM (Industrial, Scientific and Medical), με κύρια μηχανή συχνοτήτων στην Ευρώπη τα 868MHz [7].

Η τεχνολογία LoRa παρέχει σημαντική αντοχή σε παρεμβολές, μιας και χρησιμοποιεί προσαρμοστικό ρυθμό μετάδοσης (Adaptive Data Rate - ADR), ενώ παράλληλα παρουσιάζει και υψηλή ευαισθησία δεκτών, γεγονότα που επιτρέπουν την επικοινωνία ακόμα και σε συνθήκες με μεγάλο θόρυβο από το περιβάλλον. Το εύρος ζώνης που χρησιμοποιείται (Bandwidth, BW) είναι συνήθως 125kHz, 250kHz ή 500kHz, ανάλογα με τις ανάγκες της εφαρμογής. Παράλληλα, η διαμόρφωση χρησιμοποιεί διαφορετικούς παράγοντες εξάπλωσης (Spreading Factors, SF) από 7 έως 12, που επηρεάζουν τον ρυθμό μετάδοσης δεδομένων και την εμβέλεια του σήματος [16].



Εικόνα 2.2: Μονέλο OSI σε αντιστοίχιση με τα LoRa και LoRaWAN επίπεδα.

[3, 10, 5, 7]

### 2.3.2 Ραδιοφωνική διάδοση σε αστικό περιβάλλον

Στο αστικό περιβάλλον, η αξιοπιστία μιας ασύρματης ζεύξης LoRa επηρεάζεται καθοριστικά από φαινόμενα ραδιοδιάδοσης όπως η απώλεια διαδρομής (path loss), η ζώνη Fresnel, η πολυδιαδρομική διάδοση (multipath propagation) και το φαινόμενο Doppler. Παρακάτω παρουσιάζονται αυτά τα φαινόμενα με έμφαση στη φυσική τους ερμηνεία και τη σημασία τους για το LoRa, καθώς και ένα αριθμητικό παράδειγμα υπολογισμού του ισοζυγίου ζεύξης (link budget) σε αστικές συνθήκες.

#### Απώλεια Διαδρομής (Path Loss)

Η απώλεια διαδρομής εκφράζει την εξασθένηση της ισχύος του σήματος καθώς αυτό διαδίδεται μέσω του χώρου. Σε ελεύθερο χώρο (χωρίς εμπόδια), η απώλεια διαδρομής αυξάνεται με την απόσταση και τη συχνότητα σύμφωνα με το θεμελιώδες μοντέλο Friis. Η εξίσωση της ελεύθερης διαδρομής σε μορφή λογαριθμικής ( $dB$ ) δίνεται από:

$$L_{FS}(dB) = 32.45 + 20 \log_{10}(d_{km}) + 20 \log_{10}(f_{MHz}) \quad (2.1)$$

όπου  $d_{km}$  η απόσταση σε χιλιόμετρα και  $f_{MHz}$  η συχνότητα σε  $MHz$ . Για παράδειγμα, σε συχνότητα  $868MHz$  και απόσταση  $2km$  (σενάριο ζεύξης LoRa στην Ευρώπη), η απώλεια ελεύθερου χώρου είναι:

$$L_{FS} = 32.45 + 20 \log_{10}(2) + 20 \log_{10}(868) \approx 97.24dB$$

Αυτό σημαίνει ότι το σήμα εξασθενεί κατά περίπου  $97dB$  λόγω διάδοσης σε ελεύθερο χώρο. Σε πραγματικές αστικές συνθήκες όμως, η απώλεια διαδρομής είναι σημαντικά μεγαλύτερη από την ιδανική περίπτωση ελεύθερου χώρου. Κτίρια, τοίχοι και γενικά τα εμπόδια σκιάζουν και διαθλούν το σήμα, με αποτέλεσμα να υπάρχουν πρόσθετες απώλειες (shadowing, diffraction losses) [11].

Εμπειρικά μοντέλα διάδοσης για πόλεις (όπως το μοντέλο Okumura-Hata) τυπικά προβλέπουν εκθέτη απωλειών μεγαλύτερο από 2 (συχνά 2.7-4 ανάλογα με την πυκνότητα των κτιρίων), γεγονός που συνεπάγεται δεκάδες  $dB$  επιπλέον της εξασθένησης, συγκριτικά με το ελεύθερο πεδίο. Για παράδειγμα, ένα μοντέλο Hata σε πυκνή αστική περιοχή μπορεί να δώσει απώλεια διαδρομής περίπου  $130-140dB$  στα  $2km$ , τιμή αρκετά υψηλότερη από τα περίπου  $97dB$  του ελεύθερου χώρου. Επομένως, το σφάλμα ζεύξης (link margin) σε αστικά περιβάλλοντα μειώνεται δραστικά αν δεν υπάρχει καθαρή οπτική επαφή. Είναι καθοριστικής σημασίας να λαμβάνεται υπόψη αυτή η πρόσθετη εξασθένιση κατά τον σχεδιασμό δικτύων LoRa, ώστε η διαθέσιμη στάθμη σήματος να παραμένει πάνω από την ευαίσθησία του δέκτη για αξιόπιστη επικοινωνία.

#### Ζώνη Fresnel και Διάθλαση

Η ζώνη Fresnel περιγράφει μια ελλειψοειδή περιοχή γύρω από την ευθεία της οπτικής επαφής μεταξύ πομπού-δέκτη, μέσα στην οποία η διάδοση συμβάλλει επιουκοδομητικά στη λήψη. Για την πρώτη ζώνη Fresnel ( $n = 1$ ), η ακτίνα  $F_1$  στο ενδιάμεσο της διαδρομής (εκεί

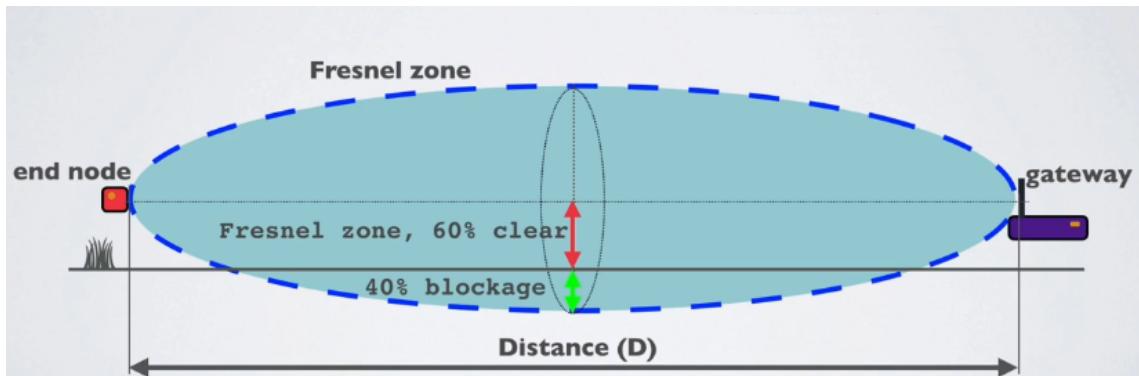
όπου η ζώνη είναι μεγαλύτερη) δίνεται από:

$$F_1 = \sqrt{\frac{\lambda d_1 d_2}{d_1 + d_2}} \quad (2.2)$$

όπου  $\lambda$  το μήκος κύματος του σήματος και  $d_1, d_2$  οι αποστάσεις του σημείου από τον πομπό και τον δέκτη αντίστοιχα. Για συχνότητα  $868MHz$  ( $\lambda \approx 0.345m$ ) και συνολική απόσταση  $2km$ , η ακτίνα της  $1^{\text{st}}$  ζώνης Fresnel στο μέσο της διαδρομής (δηλ.  $d_1 = d_2 = 1km$ ) είναι:

$$F_1 \approx \sqrt{\frac{0.345 \times 1000 \times 1000}{2000}} \approx 13m$$

Η φυσική σημασία αυτής της ζώνης είναι ότι τουλάχιστον το 60% της πρώτης ζώνης Fresnel πρέπει να είναι ελεύθερο από εμπόδια για να μην προκληθεί σημαντική πρόσθετη εξασθένηση [17]. Αν αντικείμενα (π.χ. κτίρια) παρεμβάλλονται και εισχωρούν βαθιά στη ζώνη Fresnel, το σήμα θα υποστεί διάθλαση (diffraction) γύρω από τα εμπόδια, επιφέροντας μεγάλες απώλειες πέραν της ελεύθερης διάδοσης. Σε αστικό περιβάλλον, συνήθως η ζεύξη δεν έχει καθαρή οπτική επαφή, μιας και η πρώτη ζώνη Fresnel συχνά τέμνεται από κτίρια, δέντρα ή άλλες δομές. Αυτό οδηγεί σε μη γραμμική οπτική ζεύξη (NLoS), όπου η λήψη βασίζεται σε διερχόμενα και περιθλασμένα κύματα. Το αποτέλεσμα είναι το σήμα να μειωθεί σημαντικά. Για τον σχεδιασμό δικτύου LoRa στην πόλη, συστήνεται η ανύψωση των κεραιών (π.χ. εγκατάσταση gateway σε ψηλά κτίρια) ώστε να μεγιστοποιείται η εκκαθάριση της ζώνης Fresnel και να ελαχιστοποιούνται οι απώλειες διάθλασης.



Εικόνα 2.3: Ζώνη Fresnel με 40% της κάλυψης από εμπόδια.  
[18]

### Πολυδιαδρομική Διάδοση (Multipath Propagation)

Λόγω των ανακλάσεων σε επιφάνειες όπως κτίρια, το έδαφος και άλλα εμπόδια, ένα ασύρματο σήμα μπορεί να φτάσει στον δέκτη μέσω πολλαπλών διαδρομών. Αυτή η πολυδιαδρομική διάδοση προκαλεί διαλείψεις (fading), μιας και τα σήματα από διαφορετικές διαδρομές μπορεί να φτάσουν με διαφορετική καθυστέρηση και φάση. Συνεπώς, εάν οι φάσεις τους είναι αντίθετες, ενδέχεται να επέλθει επιζήμια συμβολή, μειώνοντας σημαντικά την ισχύ του λαμβανόμενου σήματος (deep fade). Σε ένα δυναμικό περιβάλλον, ακόμη και μικρές μετακινήσεις ή αλλαγές μπορούν να μεταβάλουν το μοτίβο συμβολής, προκαλώντας

ταχεία διακύμανση του σήματος (selective fading).

Για τα δίκτυα LoRa, η πολυδιαδρομή αποτελεί κρίσιμο φαινόμενο σε αστικές περιοχές. ωστόσο η ίδια η διαμόρφωση LoRa παρουσιάζει αξιοσημείωτη αντοχή σε multipath εξασθένιση. Η διαμόρφωση Chirp Spread Spectrum (CSS) που χρησιμοποιεί το LoRa εκπέμπει το σύμβολο ως ένα ευρύ φάσμα συχνοτήτων (chirp), γεγονός που το καθιστά λιγότερο επιρρεπές σε συχνόλεκτες διαλείψεις: το φάσμα του σήματος είναι σχετικά ευρύ και λειτουργεί αποτελεσματικά σαν ένα είδος διασποράς στο πεδίο του χρόνου και της συχνότητας [19]. Σύμφωνα με τεκμηρίωση της Semtech, το φαρδύ chirp προσδίδει στο LoRa «ανοσία στην πολυδιαδρομή και στο fading, καθιστώντας το ιδανικό για αστικά και προαστιακά περιβάλλοντα όπου αυτά τα φαινόμενα κυριαρχούν» [11]. Πειραματικές μελέτες επιβεβαιώνουν αυτή την ανθεκτικότητα: ο Staniec και ο Kowal (2018) ανέφεραν ότι το LoRa παρουσιάζει αξιοσημείωτη ανοχή σε έντονες συνθήκες multipath και παρεμβολών, ιδίως στα χαμηλότερα bit-rate (μεγαλύτερους spreading factors). Συγκεκριμένα, οι μετρήσεις τους σε θάλαμο πολλαπλών ανακλάσεων έδειξαν πως για ένα εύρος ρυθμίσεων LoRa υφίστανται περιοχές ευαισθησίας: μια «λευκή» περιοχή όπου το σήμα είναι πρακτικά ανεπηρέαστο από multipath, μια «ανοιχτή γκρίζα» όπου το σύστημα εξακολουθεί να παρουσιάζει ανοχή στο multipath, αλλά αρχίζει να επηρεάζεται από ισχυρές παρεμβολές και μια «σκούρα γκρίζα» περιοχή όπου υπό ακραίες συνθήκες το LoRa γίνεται ευάλωτο και στα δύο φαινόμενα [19].

Παρότι το LoRa είναι εγγενώς ανθεκτικό, η πολυδιαδρομική διάδοση σε αστικά περιβάλλοντα μπορεί ακόμη να δημιουργήσει προκλήσεις. Αν οι χρονικές καθυστερήσεις ορισμένων διαδρομών πλησιάσουν τη διάρκεια συμβόλου LoRa, μπορεί να προκληθεί παρεμβολή συμβόλων (inter-symbol interference). Ωστόσο, δεδομένου ότι οι ρυθμοί μετάδοσης LoRa είναι χαμηλοί (μεγάλες διάρκειες συμβόλων ειδικά σε υψηλό SF), οι περισσότερες ανακλώμενες συνιστώσες καταφθάνουν εντός του παραθύρου ενός συμβόλου και συγχωνεύονται χωρίς να καταστρέφουν την πληροφορία. Έτσι, στην πράξη το LoRa σπανίως υφίσταται ολική απώλεια λόγω multipath, σε αντίθεση με τεχνολογίες υψηλότερου ρυθμού όπου το multipath οδηγεί σε έντονο επιλεκτικό fading. Παρ' όλα αυτά, για μέγιστη αξιοπιστία σε πόλεις, συνιστάται η επιλογή παραμέτρων που μεγιστοποιούν το link margin (π.χ. υψηλός SF που προσφέρει μεγαλύτερη ευαισθησία δέκτη) ώστε ακόμη και τυχόν βαθιές διαλείψεις να μην ρίχνουν το σήμα κάτω από το όριο λήψης.

### Ισοζύγιο Ζεύξης (Link Budget)

Το ισοζύγιο ζεύξης (link budget) αποτελεί έναν από τους πιο κρίσιμους παράγοντες στον σχεδιασμό συστημάτων ασύρματης επικοινωνίας, καθώς εκφράζει το συνολικό εύρος απωλειών που μπορεί να αντέξει το σήμα κατά τη μετάδοση, χωρίς να χάσει τη δυνατότητα αξιόπιστης λήψης. Πρακτικά, το ισοζύγιο ζεύξης υπολογίζεται ως η διαφορά μεταξύ της ισχύος εκπομπής και της ελάχιστης ισχύος που απαιτεί ο δέκτης για να λειτουργήσει αποτελεσματικά:

$$\text{Link Budget (dB)} = P_{TX}(dBm) + G_{TX}(dBi) + G_{RX}(dBi) - \text{Sensitivity}_{RX}(dBm) - \text{Losses}_{misc}(dB) \quad (2.3)$$

όπου:

- $P_{TX}$  είναι η ισχύς εξόδου του πομπού (σε dBm),

- $G_{TX}, G_{RX}$  είναι τα κέρδη των κεραιών πομπού και δέκτη αντίστοιχα (σε  $dB_i$ ),
- $Sensitivity_{RX}$  είναι η ευαισθησία του δέκτη (σε  $dBm$ ),
- $Losses_{misc}$  είναι διάφορες επιπλέον απώλειες (π.χ. λόγω καλωδίων, συνδέσεων ή περιβαλλοντικών συνθηκών).

Η υψηλή τιμή του ισοζυγίου ζεύξης υποδηλώνει ότι το σύστημα μπορεί να λειτουργήσει αξιόπιστα ακόμα και με πολύ ασθενή σήματα ή σε δύσκολες συνθήκες μετάδοσης (π.χ. απομακρυσμένες συσκευές, αστικά περιβάλλοντα με πολλά εμπόδια). Η τεχνολογία LoRa είναι γνωστή για το ιδιαίτερα υψηλό ισοζύγιο ζεύξης της, που τυπικά μπορεί να φτάσει έως και  $154dB$ , ανάλογα με τις παραμέτρους της διαμόρφωσης (κυρίως τον παράγοντα εξάπλωσης  $SF$  και το εύρος ζώνης  $BW$ ).

Η επίτευξη υψηλού ισοζυγίου ζεύξης στο LoRa προέρχεται από δύο βασικά χαρακτηριστικά :

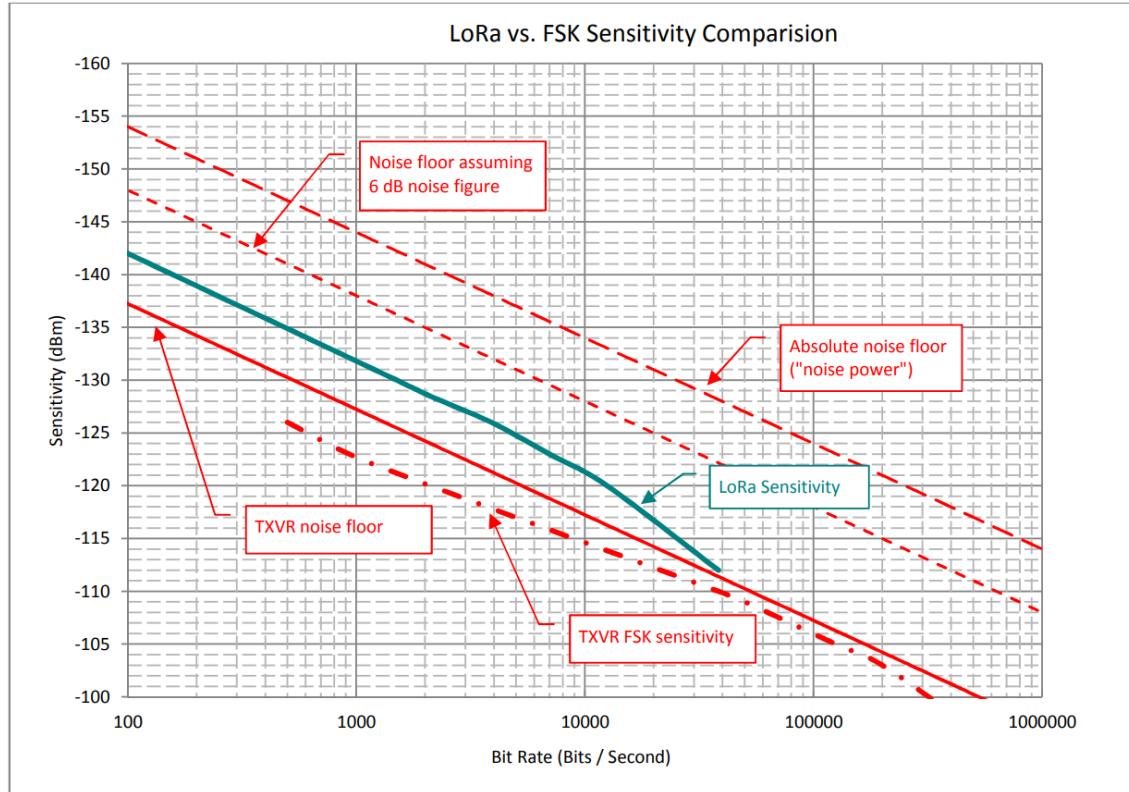
- **Υψηλή Ευαισθησία Δεκτών:** Οι δέκτες LoRa είναι σχεδιασμένοι ώστε να μπορούν να αποκωδικοποιούν σήματα πολύ χαμηλής έντασης, ακόμη και κάτω από το επίπεδο του θορύβου. Χαρακτηριστικά, η ευαισθησία των δεκτών LoRa κυμαίνεται από  $-125dBm$  (για SF7,  $BW = 125kHz$ ) έως  $-137dBm$  (για SF12,  $BW = 125kHz$ ), κάτι που επιτρέπει την λήψη εξαιρετικά ασθενών σημάτων από πολύ μεγάλες αποστάσεις [11].
- **Χαμηλός Ρυθμός Μετάδοσης και Διαμόρφωση Ευρέως Φάσματος (Spread Spectrum):** Η διαμόρφωση LoRa (Chirp Spread Spectrum, CSS) διαχέει το σήμα σε μεγάλο χρονικό διάστημα και εύρος ζώνης, αυξάνοντας την πιθανότητα αξιόπιστης λήψης του σήματος. Με τον τρόπο αυτό επεξεργάζεται ένα κέρδος επεξεργασίας (processing gain) που δίνεται περίπου από τη σχέση :

$$G_{processing} = 10 \cdot \log_{10} \left( \frac{BW}{R_{data}} \right) \quad (2.4)$$

όπου  $BW$  είναι το εύρος ζώνης μετάδοσης και  $R_{data}$  ο ρυθμός δεδομένων. Αυτό το κέρδος επεξεργασίας ενισχύει το σήμα σε σχέση με τον θόρυβο, επιτρέποντας την αποκωδικοποίηση ακόμη και υπό πολύ χαμηλές τιμές λόγου σήματος προς θόρυβο (SNR) [11].

Η τεχνολογία LoRa συγκρινόμενη με την παραδοσιακή διαμόρφωση συχνότητας (Frequency Shift Keying - FSK), η οποία χρησιμοποιείται ευρέως σε άλλες εφαρμογές ασύρματης επικοινωνίας, παρουσιάζει σημαντικά υψηλότερη ευαισθησία. Αυτό οφείλεται στο ότι η FSK απαιτεί ένα ελάχιστο θετικό SNR για αξιόπιστη αποκωδικοποίηση, ενώ η LoRa μπορεί να λειτουργήσει ακόμα και με αρνητικό SNR, με το σήμα κυριολεκτικά «θαμμένο» μέσα στον θόρυβο, όπως φαίνεται και στο Σχήμα 2.4.

Συνοψίζοντας, το υψηλό ισοζύγιο ζεύξης είναι ο κύριος λόγος που η τεχνολογία LoRa μπορεί να επιτύχει επικοινωνία σε πολύ μεγάλες αποστάσεις (έως και δεκάδες χιλιόμετρα σε ανοιχτό πεδίο), με πολύ χαμηλή ισχύ εκπομπής, καθιστώντας την ιδανική για εφαρμογές χαμηλής κατανάλωσης σε περιβάλλοντα όπου η συντήρηση και η αντικατάσταση μπαταριών είναι δύσκολη ή και αδύνατη.



Εικόνα 2.4: Σύγκριση ευαισθησίας LoRa και FSK, υπογραμμίζοντας την υπεροχή της τεχνολογίας LoRa σε συνδήκες χαμηλού λόγου σήματος προς θόρυβο (SNR).

[11]

### Φαινόμενο Doppler

Το φαινόμενο Doppler είναι η μεταβολή της συχνότητας ενός κύματος λόγω σχετικής κίνησης πομπού ή δέκτη. Στα ασύρματα δίκτυα, εάν ένας κόμβος LoRa κινείται (π.χ. αισθητήρας σε όχημα) ή αν το περιβάλλον μεταβάλει αποτελεσματικά τη συχνότητα (π.χ. ανακλώμενη διάδοση από κινούμενα αντικείμενα), η φέρουσα συχνότητα του σήματος όπως την αντιλαμβάνεται ο δέκτης μετατοπίζεται. Η μετατόπιση Doppler  $\Delta f$  προσεγγίζεται από τη σχέση:

$$\Delta f \approx \frac{v}{c} f_c, \quad (2.5)$$

όπου  $v$  η σχετική ταχύτητα πομπού-δέκτη,  $c$  η ταχύτητα του φωτός και  $f_c$  η φερουσα συχνότητα. Για παράδειγμα, στα 868MHz, αν ένας αισθητήρας κινείται με  $v = 100km/h$  ( $\approx 27.8m/s$ ), τότε η παρατηρούμενη συχνότητα μετατοπίζεται κατά:

$$\Delta f \approx \frac{27.8}{3 \times 10^8} \times 868 \times 10^6 \approx 80Hz.$$

Μια μετατόπιση περίπου 80Hz είναι αμελητέα συγκριτικά με το εύρος ζώνης του LoRa (τυπικά 125kHz) και συνεπώς δεν υποθαθμίζει την αξιοπιστία του σήματος. Γενικά, το LoRa είναι εξαιρετικά ανεκτικό στο φαινόμενο Doppler για τις συνήθεις ταχύτητες που συναντώνται σε αστικά σενάρια (π.χ. οχήματα). Η ίδια η διαμόρφωση με chirp καθιστά το σήμα ανθεκτικό σε μικρές συχνές αποκλίσεις: μια μόνιμη μετατόπιση συχνότητας λόγω Doppler απλώς μετα-

φράζεται σε μια μικρή χρονική ολίσθηση του τοπικού χρονοδιαγράμματος αποδιαμόρφωσης, κάτι που ο δέκτης LoRa μπορεί να αντιμετωπίσει χωρίς σημαντικές απώλειες απόδοσης. Στην πράξη, αυτό σημαίνει ότι δεν απαιτούνται κρύσταλλοι υψηλής ακρίβειας για τους ταλαντωτές και ότι το LoRa λειτουργεί αξιόπιστα ακόμη και σε κινητικές εφαρμογές, όπως σε αισθητήρες πίεσης ελαστικών, συστήματα διοδίων ή συσκευές σε μέσα μεταφοράς [11].

Φυσικά, σε ακραίες περιπτώσεις πολύ υψηλών ταχυτήτων (πέρα από τα αστικά δεδομένα, π.χ. σε δορυφορικές ζεύξεις LoRa) το φαινόμενο Doppler μπορεί να γίνει υπολογίσιμο, απαιτώντας τεχνικές διόρθωσης (frequency offset compensation). Όμως για επίγειες αστικές επικοινωνίες LoRa, ακόμη και ταχύτητες της τάξης των  $100 - 200 \text{ km/h}$  μπορούν να εξυπηρετηθούν χωρίς αξιόλογη υποθάμμιση της ευαισθησίας [11]. Συνοψίζοντας, το φαινόμενο Doppler δεν αποτελεί κυρίαρχο περιορισμό στην αξιοπιστία ενός στατικού δικτύου LoRa ή με αργά κινούμενους κόμβους στην πόλη.

### Παράδειγμα Υπολογισμού Απώλειας Διαδρομής και Link Budget

Στην ενότητα αυτή παρουσιάζεται ένα αριθμητικό παράδειγμα που συνδυάζει τον υπολογισμό της απώλειας διαδρομής και του ισοζυγίου ζεύξης (link budget) για μια χαρακτηριστική σύνδεση LoRa σε αστικό περιβάλλον. Ας θεωρήσουμε ένα σενάριο όπου:

- Συχνότητα λειτουργίας:  $f = 868 \text{ MHz}$  (ζώνη EU863-870).
- Απόσταση πομπού-δέκτη:  $d = 2 \text{ km}$  (αστικό περιβάλλον, πιθανώς χωρίς καθαρή οπτική επαφή).
- Ισχύς εκπομπής πομπού:  $P_{TX} = 14 \text{ dBm}$  (τυπικό μέγιστο LoRa επιτρεπόμενο στην ΕΕ).
- Κέρδος κεραίας πομπού/δέκτη:  $G_{TX} = 0 \text{ dBi}$  (μικρή μονοπολική στον αισθητήρα),  $G_{RX} = 2 \text{ dBi}$  (κεραία gateway).
- Απώλειες καλωδίων/συνδέσεων:  $L_{misc} = 2 \text{ dB}$  (π.χ. απώλεια ομοαξονικού στον σταθμό βάσης).
- Ευαισθησία δέκτη:  $S_{min} \approx -137 \text{ dBm}$  (υψηλή ευαισθησία, π.χ. LoRa δέκτης σε  $SF = 12, BW = 125 \text{ kHz}$ ) [20].

**1. Υπολογισμός απώλειας διαδρομής:** Χρησιμοποιούμε πρώτα την εξίσωση ελεύθερου χώρου. Για  $d = 2 \text{ km}$ ,  $f = 868 \text{ MHz}$ , όπως υπολογίστηκε προηγουμένως,  $L_{FS} \approx 97.4 \text{ dB}$ . Σε αστικό περιβάλλον χωρίς οπτική επαφή, θα προσθέσουμε μια επιπλέον απώλεια λόγω σκίασης/διάθλασης. Ας υποθέσουμε μια συντηρητική επιπλέον εξασθένηση  $L_{urban} = 20 \text{ dB}$  (λόγω κτιρίων που μερικώς φράζουν τη ζώνη Fresnel και προκαλούν διάθλαση). Έτσι, η συνολική εκτιμώμενη απώλεια διαδρομής γίνεται:

$$L_{path} = L_{FS} + L_{urban} \approx 97.4 + 20 = 117.4 \text{ dB}$$

**2. Λήψη και Ισοζύγιο Ζεύξης:** Το ισοζύγιο ζεύξης λαμβάνει υπόψη όλα τα κέρδη και τις απώλειες από τον πομπό έως τον δέκτη. Η ισχύς που φτάνει στον δέκτη ( $P_{RX}$  σε  $\text{dBm}$ ) δίνεται από:

$$P_{RX} = P_{TX} + G_{TX} + G_{RX} - L_{path} - L_{misc}$$

Αντικαθιστώντας τις αριθμητικές τιμές:

$$P_{RX} = 14dBm + 0dB + 2dB - 117.4\ dB - 2dB$$

$$P_{RX} \approx -103.4dBm$$

Η λαμβανόμενη ισχύς εκτιμάται περίπου  $-103.4dBm$ .

**3. Σύγκριση με ευαισθησία δέκτη:** Δεδομένου ότι ο δέκτης LoRa (με  $SF = 12$ ) μπορεί να ανιχνεύσει σήματα έως και  $S_{min} \approx -137dBm$ , το συγκεκριμένο σενάριο παρουσιάζει ένα περιθώριο ζεύξης γύρω στα  $33.6dB$  (διαφορά μεταξύ  $-103.4dBm$  και  $-137dBm$ ). Αυτό το περιθώριο είναι πολύ άνετο, υπερκαλύπτει τυχόν επιπλέον απώλειες λόγω πιο έντονου fading ή παρεμβολών, εξασφαλίζοντας αξιόπιστη επικοινωνία. Σημειώνεται ότι το υπολογισθέν  $L_{path}$  περιείχε ήδη ένα αποθεματικό  $20dB$  για αστικό περιβάλλον. Στην πράξη, αν η ζεύξη είχε οπτική επαφή (LoS) το περιθώριο θα ήταν ακόμη μεγαλύτερο.

**4. Επίδραση παραμέτρων LoRa:** Αξίζει να τονιστεί ότι η ευαισθησία  $-137dBm$  αντιστοιχεί σε διαμόρφωση LoRa με τον πιο παρατεταμένο χρόνο συμβόλου ( $SF12$ ,  $BW = 125kHz$ ) [20]. Εάν χρησιμοποιούταν μια ταχύτερη διαμόρφωση (π.χ.  $SF7$  με ευαισθησία περί τα  $-123dBm$ ), το περιθώριο ζεύξης θα μειωνόταν (περίπου  $20dB$  λιγότερο ευαισθητος δέκτης) αλλά πιθανώς να παρέμενε επαρκές για  $2km$ . Στην περίπτωσή μας, με  $SF12$ , το πολύ μεγάλο link margin των  $33+ dB$  υποδηλώνει ότι η ζεύξη θα εξακολουθούσε να λειτουργεί ακόμα κι αν η απώλεια διαδρομής ήταν σημαντικά μεγαλύτερη (π.χ. περίπου μέχρι  $150dB$  συνολικά). Πράγματι, τα σύγχρονα LoRa transceivers υποστηρίζουν μέγιστο link budget της τάξης των  $155 - 170dB$ , ικανό να καλύψει αποστάσεις πολλών δεκάδων χιλιομέτρων σε ιδινές συνθήκες. Στο δικό μας σενάριο, μια απώλεια περίπου  $117dB$  είναι αρκετά χαμηλή συγκριτικά με το διαθέσιμο link budget (περίπου  $154dB$  με  $168dB$ , ανάλογα τον δέκτη), εξηγώντας γιατί οι ζεύξεις LoRa μπορούν να επιτύχουν αξιόπιστη επικοινωνία ακόμα και σε αστικό περιβάλλον με διάφορες προκλήσεις διάδοσης [16].

**Συμπέρασμα του παραδείγματος:** Με τις παραπάνω παραμέτρους, η ζεύξη LoRa στα  $2km$  όχι μόνο «κλείνει» (δηλαδή το σήμα υπερβαίνει το κατώφλι ευαισθησίας του δέκτη), αλλά διαθέτει και σημαντικό περιθώριο αξιοπιστίας. Αυτό το περιθώριο μπορεί να απορροφήσει επιπλέον απώλειες από φαινόμενα όπως εξασθένηση multipath, μελλοντική υποβάθμιση σήματος λόγω παρεμβολών ή μείωση ισχύος μπαταρίας του πομπού. Δείχνει επίσης τη σπουδαιότητα του link budget, αφού, συνδυάζοντας υψηλή ευαισθησία δέκτη, επαρκή ισχύ εκπομπής και κεραίες με μικρά έστω κέρδη, το LoRa πετυχαίνει μεγάλες εμβέλειες. Σε ακραίες αστικές συνθήκες (π.χ. πολύ πυκνό αστικό τοπίο, εσωτερικό κτιρίων), το περιθώριο αυτό θα μειωθεί, ωστόσο η εγγενής ανθεκτικότητα του πρωτοκόλλου (λόγω του χαμηλού ρυθμού μετάδοσης και του spread spectrum) επιτρέπει στο LoRa να διατηρεί επικοινωνία ακόμη και εκεί όπου άλλα συστήματα υψηλότερης συχνότητας ή ταχύτητας αποτυγχάνουν.

### 2.3.3 Chirp Spread Spectrum και η υλοποίησή του στη LoRa

Η τεχνολογία LoRa χρησιμοποιεί τη διαμόρφωση φάσματος διασποράς με chirp (Chirp Spread Spectrum, CSS). Η τεχνική αυτή αναπτύχθηκε αρχικά τη δεκαετία του 1940 για εφαρμογές ραντάρ και έκτοτε έχει χρησιμοποιηθεί σε στρατιωτικά και ασφαλή συστήματα

επικοινωνιών [21]. Τα τελευταία χρόνια γνωρίζει ευρεία εφαρμογή σε ασύρματες επικοινωνίες δεδομένων, λόγω των σχετικά χαμηλών απαιτήσεων ισχύος και της έμφυτης ανθεκτικότητάς της έναντι παραγόντων υποβάθμισης του καναλιού, όπως η πολλαπλή διάδοση, το fading, το φαινόμενο Doppler κ.λπ. που αναφέρθηκαν προηγουμένως [11]. Μάλιστα, ένα φυσικό στρώμα βασισμένο σε CSS υιοθετήθηκε από το πρότυπο IEEE 802.15.4a για ασύρματα προσωπικά δίκτυα χαμηλού ρυθμού (LR-WPAN) που απαιτούν μεγαλύτερη εμβέλεια και κινητικότητα, ως εναλλακτική του DSSS με O-QPSK [21].

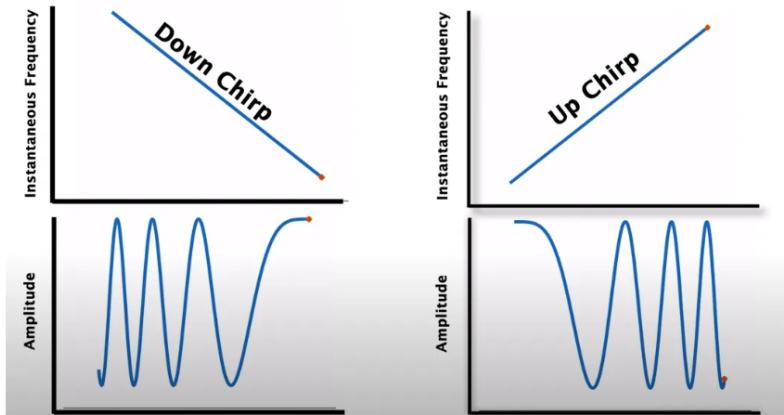
Σε αυτό το σχήμα διαμόρφωσης, κάθε σύμβολο μεταδίδεται ως ένα ημιτονοειδές σήμα (σήμα chirp). Συνεπώς, αντί για χρήση ψευδοτυχαίων κωδίκων διάχυσης φάσματος όπως στο DSSS, στη LoRa το φάσμα διασπείρεται μέσω σημάτων chirp (παλμών των οποίων η συχνότητα μεταβάλλεται γραμμικά με τον χρόνο). Ένα τέτοιο chirp σαρώνει ολόκληρο το διαθέσιμο εύρος ζώνης κατά τη διάρκεια ενός συμβόλου. Ένα σημαντικό πλεονέκτημα αυτής της μεθόδου είναι ότι τυχόν διαφορές χρονισμού ή/και συχνότητας μεταξύ πομπού και δέκτη αντισταθμίζονται αυτόματα (καθώς και οι δύο υφίστανται την ίδια μετατόπιση), γεγονός που απλοποιεί σημαντικά τον σχεδιασμό και τη διαδικασία συγχρονισμού του δέκτη [11]. Το φάσμα συχνότητας του chirp καθορίζεται από το εύρος ζώνης  $BW$  (Bandwidth) του σήματος (στην ουσία, το εύρος ζώνης του chirp είναι ίσο με το εύρος ζώνης του εκπεμπόμενου σήματος). Η επιθυμητή πληροφορία (τα δεδομένα) «τεμαχίζεται» σε πολύ υψηλότερο ρυθμό (chips) και κάθε σύμβολο μεταδίδεται ως ένα chirp μέσα στο συγκεκριμένο εύρος ζώνης, ενσωματώνοντας τα δεδομένα στην αρχική φάση ή συχνότητα του chirp.

Στην πράξη, ένα up-chirp (που η συχνότητά του αυξάνεται γραμμικά) χρησιμοποιείται για την εκπομπή των συμβόλων, ενώ η LoRa ορίζει και ειδικά down-chirps (φθίνουσας συχνότητας) για σήματα συγχρονισμού και τερματισμού. Η κυματομορφή ενός ιδανικού up-chirp μπορεί να περιγραφεί μαθηματικά ως σήμα συχνότητας που μεταβάλλεται με σταθερό ρυθμό. Για παράδειγμα, ένα απλοποιημένο μοντέλο up-chirp ξεκινώντας από συχνότητα  $f_0$  μπορεί να γραφεί ως:

$$s(t) = \cos(2\pi f_0 t + \pi \frac{BW}{T_s} t^2), \quad 0 \leq t < T_s \quad (2.6)$$

όπου  $T_s$  η διάρκεια του συμβόλου (chirp). Στο διάστημα αυτό, η στιγμιαία συχνότητα του  $s(t)$  αυξάνεται γραμμικά από  $f_0$  έως  $f_0 + BW$ . Χάρη στην τεχνική CSS, οι δέκτες LoRa μπορούν να ανιχνεύσουν σήματα έως και 19.5dB κάτω από το επίπεδο θορύβου του καναλιού, αξιοποιώντας διαδικασίες συσχέτισης (correlation/demodulation) που συμπιέζουν το διεσπαρμένο σήμα πίσω στο αρχικό φάσμα του [21]. Συγκριτικά με τα συστήματα DSSS που χρησιμοποιούν ακολουθίες ψευδοθορύβου (όπως π.χ. το 802.11 ή το UMTS), η LoRa χρησιμοποιεί chirp pulses αντί για ψευδοτυχαίους κώδικες για τη διασπορά [21]. Το εκπεμπόμενο σήμα LoRa έχει συνεχή χαρακτηριστική με σταθερή περιβάλλουσα διαμόρφωση (constant envelope FM), η οποία αυξάνεται ή μειώνεται μονοτονικά εντός του διαθέσιμου φάσματος. Αυτή η προσέγγιση επιτυγχάνει την ίδια λειτουργία διάχυσης φάσματος, με χαμηλή πολυπλοκότητα και χωρίς να απαιτείται ακριβής γεννήτρια χρονισμού για μακρές ακολουθίες κώδικα, όπως συνέβαινε σε DSSS εφαρμογές (π.χ. GPS) [11]. Συνολικά, η CSS διαμόρφωση της LoRa παρέχει μια ανθεκτική λύση spread spectrum με χαμηλό κόστος και κατανάλωση, κατάλληλη για δίκτυα IoT μεγάλης εμβέλειας.

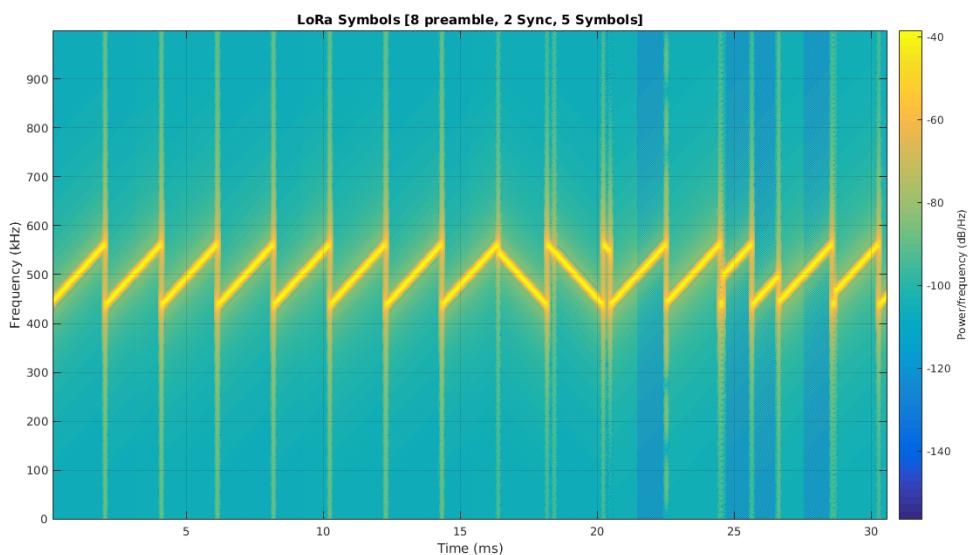
Η μετάδοση ενός πακέτου LoRa αρχίζει με έναν προκαθορισμένο πρόλογο (preamble)



Εικόνα 2.5: Παράλληλη σύγκριση ενός down-chirp (αριστερά) και ενός up-chirp (δεξιά), που δείχνει τη γραμμική μείωση/αύξηση της σπιγμαίας συχνότητας (πάνω) συνοδευόμενη από το αντίστοιχο σήμα στο πεδίο του χρόνου, όπου η απόσταση των κυμάτων εκτείνεται/συμπιέζεται καθ' όλη τη διάρκεια του συμβόλου.

[22]

από διαδοχικά up-chirps που επιτρέπουν στον δέκτη να αντιληφθεί την παρουσία σήματος και να συγχρονίσει τη συχνότητα και το ρολόι του. Τυπικά χρησιμοποιούνται 8 σύμβολα προοιμίου (στην Ευρώπη), τα οποία ακολουθούνται από 2 ειδικά down-chirps που σηματοδοτούν το τέλος του προοιμίου και βοηθούν στον ακριβή συγχρονισμό φάσης του δέκτη [23]. Μετά το προοίμιο, έπονται τα chirps που μεταφέρουν τα ωφέλιμα δεδομένα, καθένα εκ των οποίων έχει τροποποιηθεί κυκλικά ως προς τη φάση ώστε να αντιστοιχεί σε μια συγκεκριμένη ακολουθία bits.

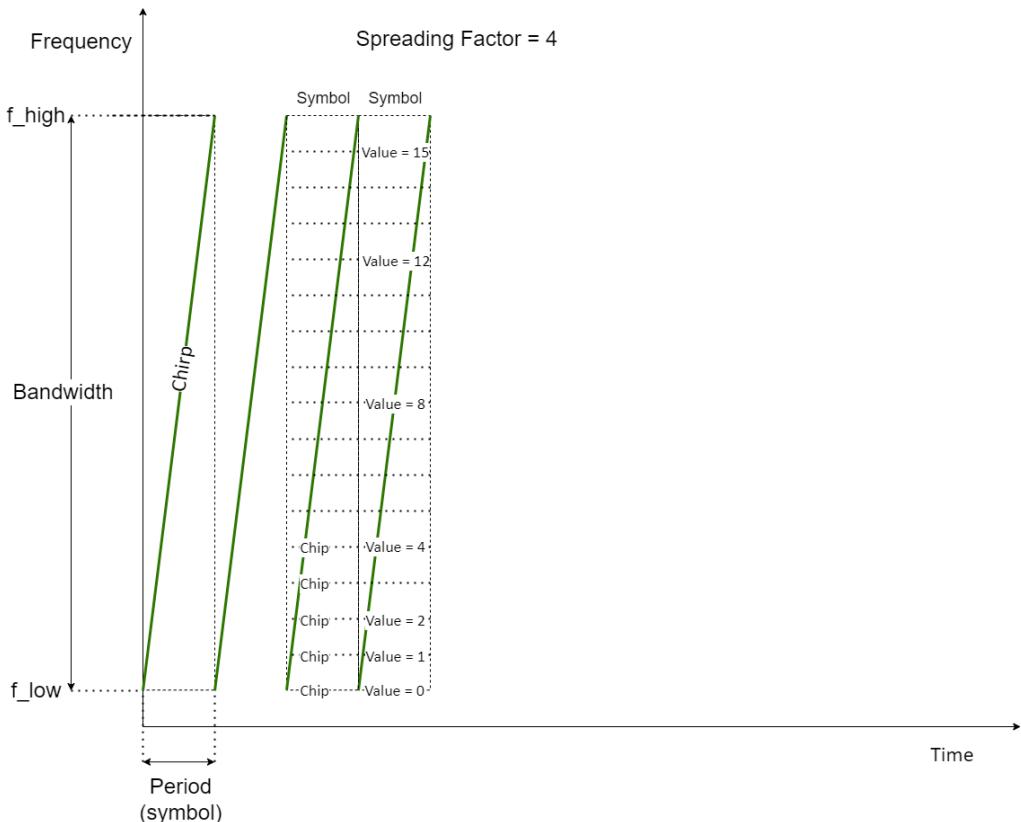


Εικόνα 2.6: Φασματογράφημα σήματος LoRa που παρουσιάζει 8 αρχικά up-chirps προοιμίου, 2 down-chirps συγχρονισμού και ακολουθία 5 chirp με κωδικοποιημένα δεδομένα (διαφορετική κυκλική μετατόπιση σε κάθε σύμβολο). Το σήμα σαρώνει πλήρως ένα εύρος ζώνης 125kHz με γραμμικά αυξανόμενη ή μειούμενη συχνότητα σε κάθε chirp.

[23]

### 2.3.4 Παράγοντας εξάπλωσης και ευαισθησία δέκτη

Όπως έχει ήδη φανεί, βασική παράμετρος στη διαμόρφωση του σήματος LoRa είναι ο Παράγοντας Εξάπλωσης (Spreading Factor, SF). Ο SF ορίζει τον βαθμό διασποράς του σήματος. Ουσιαστικά ισούται με τον αριθμό των bits που κωδικοποιούνται σε κάθε σύμβολο. Μια μετάδοση με  $SF = n$  σημαίνει ότι κάθε σύμβολο αντιστοιχεί σε  $n$  bits ωφέλιμης πληροφορίας, τα οποία διασπείρονται σε ένα chirp διάρκειας  $2^n$  chips.



Εικόνα 2.7: Σχηματική απεικόνιση της κυματομορφής Chirp Spread Spectrum στο LoRa για Spreading Factor (4 bits). Το διάγραμμα δείχνει πώς κάθε σύμβολο σαρώνει γραμμικά το εύρος ζώνης από  $f_{low}$  σε  $f_{high}$ , καθώς και την αντιστοίχιση των chips σε διακριτές τιμές (0-15) μέσα σε ένα σύμβολο.

[24]

Συνολικά, υπάρχουν διαθέσιμες 6 διακριτές τιμές (τυπικά  $SF = 7$  έως  $SF = 12$ ) για το LoRa PHY<sup>2</sup>. Μεγαλύτερος SF συνεπάγεται περισσότερα chips ανά σύμβολο και άρα μεγαλύτερη διάρκεια συμβόλου, κάτι που μειώνει τον ρυθμό μετάδοσης δεδομένων, αλλά αυξάνει το processing gain και την ευαισθησία του δέκτη. Αντίθετα, μικρότερος SF δίνει ταχύτερη μετάδοση (περισσότερα symbol/s) αλλά με χαμηλότερη επεξεργαστική ενίσχυση και συνεπώς μικρότερη ακτίνα αξιόπιστης επικοινωνίας. Στην πράξη, κάθε αύξηση του SF κατά 1 περίπου μονάδα διπλασιάζει τη διάρκεια του συμβόλου (για σταθερό εύρος ζώνης) με αποτέλεσμα να υποδιπλασιάζεται ο ρυθμός δεδομένων και να απαιτείται υψηλότερο  $E_b/N_0$  (SNR per bit) για σωστή λήψη (αν δεν αλλάζει η ισχύς) [12]. Από την άλλη, ένα μεγαλύτερο SF

<sup>2</sup>Υπάρχει επίσης  $SF = 6$  σε ορισμένες υλοποιήσεις, αλλά χρησιμοποιείται μόνο σε ειδικές περιπτώσεις και με διαφοροπιθημένο πρωτόκολλο διαμόρφωσης.

επιτρέπει στο σήμα να ταξιδέψει μακρύτερα, αφού μπορεί να ανακτηθεί σωστά ακόμα και με πολύ χαμηλότερο  $SNR$  στον δέκτη λόγω του υψηλού κέρδους διασποράς (π.χ. ένα πακέτο με  $SF = 12$  μπορεί να φτάσει αποδοτικά πιο μακριά από ένα με  $SF = 7$ , υπό τις ίδιες λοιπές συνθήκες [12]). Η επιλογή του  $SF$  σε δίκτυα LoRaWAN γίνεται συνήθως δυναμικά, μέσω του μηχανισμού Adaptive Data Rate, ώστε να επιτευχθεί ισορροπία μεταξύ ρυθμού μετάδοσης και αξιοπιστίας/απόστασης για κάθε συσκευή.

Ένα ιδιαίτερα σημαντικό χαρακτηριστικό της LoRa είναι ότι τα σήματα που χρησιμοποιούν διαφορετικούς  $SF$  (σε κοινό κανάλι συχνότητας και εύρος ζώνης) είναι σχεδόν ορθογώνια μεταξύ τους [16]. Αυτό σημαίνει ότι μια πύλη (gateway) LoRa μπορεί να λαμβάνει και να διαχωρίζει ταυτόχρονα πολλαπλές μεταδόσεις στην ίδια συχνότητα, εφόσον αυτές γίνονται με διαφορετικούς spreading factors, χωρίς ουσιαστική αλληλοπαρεμβολή. Η ορθογωνικότητα προκύπτει διότι τα διασπειρόμενα σήματα με διαφορετικό  $SF$  έχουν πολύ χαμηλή συσχέτιση: η εγκάρσια συσχέτιση των αντίστοιχων σειρών chips τείνει στο μηδέν, με αποτέλεσμα ένα πακέτο π.χ.  $SF = 10$  να εμφανίζεται ως θόρυβος στον δέκτη που «ακούει»  $SF = 7$  και αντιστρόφως [25]. Αυτό επιτρέπει την ταυτόχρονη εξυπηρέτηση πολλών κόμβων στον ίδιο δίαυλο, ουσιαστικά πολλαπλασιάζοντας τη χωρητικότητα του καναλιού, αφού συσκευές με διαφορετικό  $SF$  δεν μετρούν ως collision μεταξύ τους στο επίπεδο PHY. Η ιδιότητα αυτή αξιοποιείται από το πρωτόκολλο LoRaWAN για τον έλεγχο της συμφόρησης: το δίκτυο μπορεί να αναθέτει υψηλότερους  $SF$  σε απομακρυσμένες ή δυσμενείς συσκευές και χαμηλότερους  $SF$  σε συσκευές με καλό σήμα, ώστε όλες να γίνονται αξιόπιστα αντιληπτές από τον δέκτη, μοιράζοντας το κανάλι χωρίς ανταγωνισμό [12].

Σημειώνεται ότι η ορθογωνικότητα μεταξύ διαφορετικών  $SF$  ισχύει αυστηρά μόνο όταν χρησιμοποιείται το ίδιο  $BW$ . Αν δύο μεταδόσεις χρησιμοποιούν διαφορετικό εύρος ζώνης και κατάλληλους  $SF$  τέτοιους ώστε να έχουν τον ίδιο ρυθμό chips, τότε οι γραμμικές σαρώσεις συχνότητας (chirps) θα έχουν ίδια «κλίση» και τα δύο σήματα δεν θα διαχωρίζονται καλά. Για παράδειγμα, ένας συνδυασμός  $SF = 7$  με  $BW = 125kHz$  και ένας άλλος με  $SF = 9$  και  $BW = 250kHz$  παράγουν το ίδιο chip rate ( $R_c$ ) και ουσιαστικά το ίδιο chirp rate (ρυθμό μεταβολής συχνότητας), άρα τα σήματα δεν είναι ορθογώνια μεταξύ τους. Στην πράξη αυτό αποφεύγεται διότι κάθε κανάλι LoRa ορίζεται από συγκεκριμένο εύρος ζώνης (π.χ.  $125kHz$ ) και μόνο ο  $SF$  μεταβάλλεται. Όλοι οι κόμβοι σε ένα συγκεκριμένο κανάλι χρησιμοποιούν το ίδιο  $BW$  (συνήθως  $125kHz$  στην Ευρώπη), διασφαλίζοντας την ορθογωνικότητα μεταξύ των διαθέσιμων SF7-SF12.

Spreading Factor	Chips per symbol	$SNR_{limit}(dB)$
7	128	-7.5
8	256	-10
9	512	-12.5
10	1024	-15
11	2048	-17.5
12	4096	-20

Πίνακας 2.2: Πίνακας παραμέτρων Spreading Factor και αντίστοιχων ορίων SNR.

### Ευαισθησία δέκτη

Η ευαισθησία του δέκτη ενός συστήματος LoRa είναι το ελάχιστο επίπεδο ισχύος στο οποίο μπορεί να ανιχνευθεί σωστά το σήμα, με βάση ένα συγκεκριμένο SNR. Η ευαισθησία υπολογίζεται από τη σχέση:

$$S(dBm) = -174 + 10 \log_{10}(BW) + NF + SNR_{limit} \quad (2.7)$$

όπου:

- $-174 dBm/Hz$ : η θερμική πυκνότητα θορύβου στους  $25^{\circ}C$ ,
- $BW$ : το εύρος ζώνης σε  $Hz$ ,
- $NF$ : ο συντελεστής θορύβου Noise Figure του δέκτη (συνήθως περίπου  $6dB$ , προκαθορισμένο αναλόγως με το hardware),
- $SNR$ : το όριο  $SNR$  του συγκεκριμένου  $SF$ .

Ο Πίνακας 2.3 παρουσιάζει τα όρια  $SNR$  και την αντίστοιχη ευαισθησία του δέκτη για διαφορετικούς παράγοντες εξάπλωσης, με τυπική τιμή  $NF = 6dB$  και  $BW = 125kHz$ .

Spreading Factor	$SNR_{limit}(dB)$	$S_{(sensitivity)}(dBm)$
7	-7.5	-125
8	-10	-127
9	-12.5	-130
10	-15	-132
11	-17.5	-135
12	-20	-137

Πίνακας 2.3: 'Όρια  $SNR$  και ευαισθησία δέκτη για διαφορετικά Spreading Factors, με  $BW = 125kHz$  και  $NF = 6dB$ .

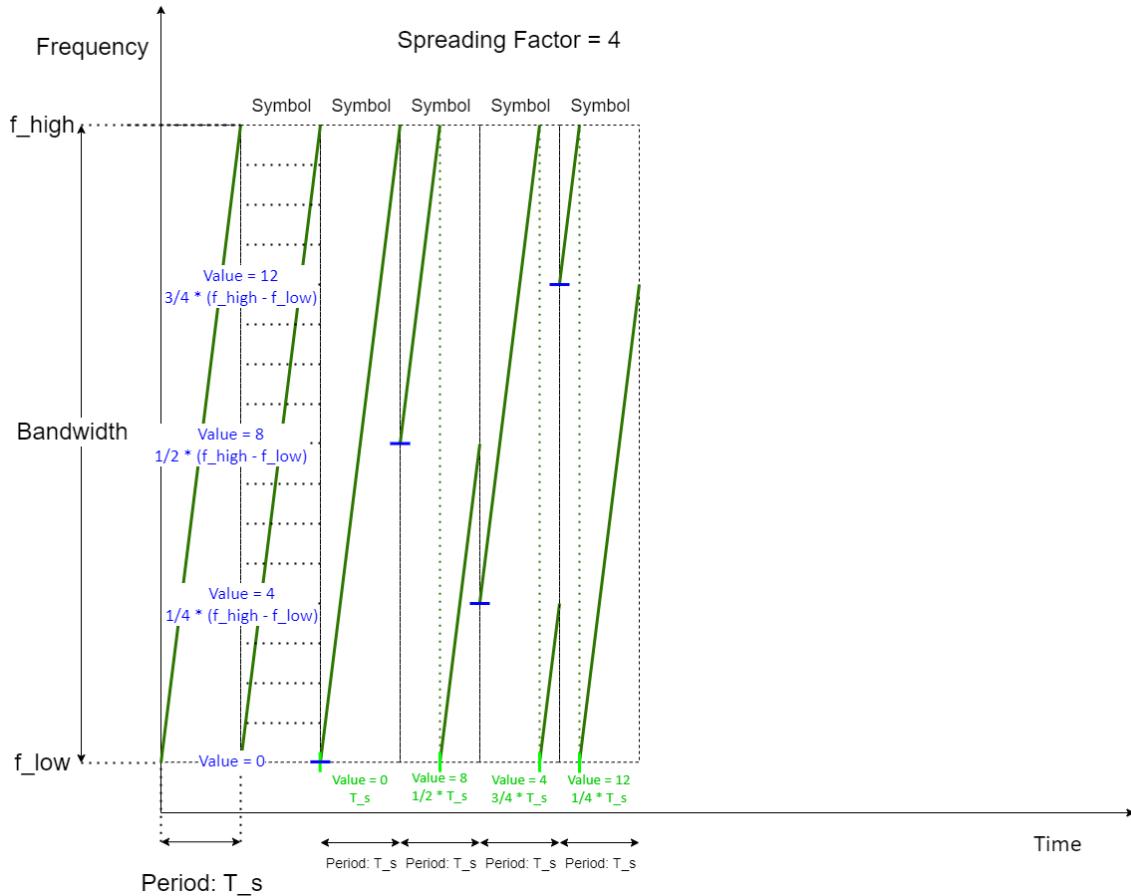
Από τον παραπάνω πίνακα, φαίνεται ότι η αύξηση του  $SF$  βελτιώνει σημαντικά την ευαισθησία, επιτρέποντας την ανίχνευση σημάτων σε χαμηλότερα επίπεδα ισχύος, κάτι που είναι καθοριστικό για τη μεγιστοποίηση της εμβέλειας σε δίκτυα LoRaWAN.

### 2.3.5 Ρυθμός συμβόλων, Chips, Bitrate και εξισώσεις

Το μήκος του συμβόλου στη LoRa (η διάρκεια ενός chirp) εξαρτάται άμεσα από τον παράγοντα εξάπλωσης και το εύρος ζώνης. Συγκεκριμένα, η χρονική διάρκεια  $T_s$  ενός συμβόλου ισούται με τον αριθμό των chips ανά σύμβολο δια το  $BW$ . Επειδή κάθε σύμβολο αποτελείται από  $2^{SF}$  chips, προκύπτει:

$$T_s = \frac{2^{SF}}{BW} \quad (2.8)$$

όπου  $BW$  το χρησιμοποιούμενο εύρος ζώνης (σε  $Hz$ ) [11]. Για παράδειγμα, με  $SF = 7$  και  $BW = 125kHz$ , έχουμε  $T_s = 2^7 / 125000 \approx 1.024ms$ , ενώ με  $SF = 12$  (ίδιο  $BW$ )



Εικόνα 2.8: Απεικόνιση κυματομορφών Chirp Spread Spectrum στο LoRa για Spreading Factor 4, με ένδειξη των διακριτών τιμών (0, 4, 8, 12) στην αντίστοιχη χρονική θέση μέσα στο σύμβολο  $T_s$ . Τα μπλε σημάδια δείχνουν το κλάσμα της περιόδου συμβόλου στο οποίο εκπέμπεται κάθε τιμή, ενώ η πράσινη σήμανση αντίστοιχει στον αριθμό του chip.

[24]

$T_s = 2^{12}/125000 \approx 32.768ms$  (δηλαδή 32 φορές μεγαλύτερο). Ο ρυθμός συμβόλων  $R_s$  (symbols per second) είναι απλά το αντίστροφο του  $T_s$ :

$$R_s = \frac{1}{T_s} = \frac{BW}{2^{SF}} \quad (2.9)$$

Όταν το SF αυξάνεται, ο ρυθμός συμβόλων μειώνεται εκθετικά (κάθε αύξηση SF κατά 1  $\Rightarrow R_s$  στο μισό). Η έννοια του chip αντίστοιχει στο μικρότερο χρονικό βήμα εντός ενός συμβόλου (με άλλα λόγια είναι πρακτικά το ελάχιστο διάστημα φάσης του chirp στο οποίο κωδικοποιείται πληροφορία. Με  $2^{SF}$  chips ανά σύμβολο και  $R_s = BW/2^{SF}$ , μπορούμε να δούμε ότι ο ρυθμός chips  $R_c$  ισούται με:

$$R_c = R_s \cdot 2^{SF} = BW \quad (2.10)$$

Το αποτέλεσμα αυτό σημαίνει ότι ανεξάρτητα από τον SF, το modem παράγει τον ίδιο αριθμό chips ανά δευτερόλεπτο, ίσο με το εύρος ζώνης. Για παράδειγμα, σε  $BW = 125kHz$ , εκπέμπονται  $125.000 chips/s$  (δηλ. κάθε chip έχει διάρκεια  $8 \mu s$ ), ενώ σε  $BW = 500kHz$  εκπέμπονται  $500.000 chips/s$ , ανεξάρτητα από το SF. Αυτή η ιδιότητα συνάδει με τον ορισμό

που δίνεται στα φυλλάδια της Semtech: «ένα chip εκπέμπεται ανά  $Hz$  ανά δευτερόλεπτο» [11].

Εφόσον κάθε σύμβολο αντιστοιχεί σε  $SF$  bits (πριν την προσθήκη κώδικα διόρθωσης), μπορούμε να εκφράσουμε τον ακαθάριστο ρυθμό bit της διαμόρφωσης LoRa. Χωρίς χρήση FEC, κάθε σύμβολο φέρει  $SF$  bits πληροφορίας και μεταδίδεται σε χρόνο  $T_s$ , άρα ο ρυθμός bit θα ήταν  $R_b = SF \cdot R_s = SF \cdot \frac{BW}{2^{SF}}$ . Στην LoRa όμως εφαρμόζεται επιπρόσθετα ένας κώδικας Forward Error Correction (FEC) τύπου Hamming, που εισάγει πλεονάζοντα bits για διόρθωση λαθών. Ο βαθμός κωδικοποίησης εκφράζεται με λόγο  $4/(4 + CR)$ , όπου  $CR$  είναι ένας ακέραιος 1-4 (π.χ.  $CR = 1$  αντιστοιχεί σε code rate  $4/5$ ,  $CR = 4$  σε  $4/8$  κ.ο.κ.). Ο καθαρός ρυθμός bit (ονομαστικός, net bit rate) λαμβάνοντας υπόψη την πλεονάζουσα κωδικοποίηση, δίνεται από την εξίσωση:

$$R_b = SF \cdot \frac{4}{4 + CR} \cdot \frac{BW}{2^{SF}} \quad (2.11)$$

όπου τα  $SF$  και  $CR$  ορίζονται όπως παραπάνω [11]. Μπορούμε να ορίσουμε για ευκολία έναν παράγοντα  $RateCode = \frac{4}{4+CR}$  (π.χ.  $RateCode = 0.8$  για  $CR = 1$ , ή  $0.5$  για  $CR = 4$ ). Τότε η σχέση γίνεται:

$$R_b = SF \cdot RateCode \cdot \frac{BW}{2^{SF}} \quad (2.12)$$

Από τις παραπάνω σχέσεις προκύπτει ότι για δεδομένο  $BW$  και  $CR$ , η αύξηση του  $SF$  μειώνει τον  $R_b$  εκθετικά. Για παράδειγμα με  $SF = 7$ ,  $BW = 125kHz$  και  $CR = 1$  (κώδικας  $4/5$ ) ο καθαρός ρυθμός είναι:

$$R_b = 7 \cdot \frac{4}{5} \cdot \frac{125000}{128} \approx 5470bit/s \quad (2.13)$$

δηλαδή περίπου  $5.47kbps$ . Με τις ίδιες παραμέτρους αλλά  $SF = 12$ , ο ρυθμός μειώνεται δραστικά:

$$R_b = 7 \cdot \frac{4}{5} \cdot \frac{125000}{4096} \approx 293bit/s \quad (2.14)$$

δηλαδή μόλις  $0.293kbps$ . Αν χρησιμοποιηθεί ο μέγιστος πλεονασμός ( $CR = 4$ , δηλ.  $4/8$ ), ο ρυθμός για  $SF = 12$  πέφτει ακόμα χαμηλότερα, περίπου  $183bit/s$ . Αυτός είναι και ο ελάχιστος ρυθμός δεδομένων σε δίκτυο LoRa για ένα μόνο κανάλι  $125kHz$ . Αντιστρόφως, η χρήση μεγαλύτερου εύρους ζώνης αυξάνει γραμμικά τον ρυθμό: για  $BW = 250kHz$  ο ρυθμός bit διπλασιάζεται (με σταθερά  $SF, CR$ ), ενώ για  $BW = 500kHz$  τετραπλασιάζεται κ.ο.κ. [12].

Σε υψηλές τιμές  $SF$ , η μεγάλη διάρκεια συμβόλου μπορεί να καταστήσει τις επικοινωνίες πιο ευάλωτες σε αστάθεια του ταλαντωτή ή σε θόρυβο φάσης. Γι' αυτό, στα πρότυπα LoRaWAN συνιστάται η ενεργοποίηση μιας ρύθμισης Low Data Rate Optimization (συντομογραφία LDRO) για  $SF \geq 11$  όταν  $BW = 125kHz$  [12]. Αυτή η ρύθμιση πρακτικά μειώνει το effective data rate εισάγοντας μικρή καθυστέρηση στη διαμόρφωση (μείωση του ρυθμού symbols/modulation), ώστε να βελτιωθεί η αξιοπιστία στον δέκτη. Στις εξισώσεις, η ενεργοποίηση του LDRO λαμβάνεται υπόψη ως ( $SF - 2$ ) στον παρονομαστή ορισμένων όρων (βλ. επόμενη ενότητα), γεγονός που αντιστοιχεί θεωρητικά σε μείωση του διαθέσιμου αριθμού bit ανά σύμβολο κατά 2 όταν το LDRO = 1.

### Βελτιστοποιήσεις CSS για LoRa

Παρά τους σχετικά χαμηλούς ρυθμούς της καθιερωμένης διαμόρφωσης LoRa, πρόσφατες ερευνητικές προσπάθειες στοχεύουν στη βελτίωση της φασματικής αποδοτικότητάς της. Για παράδειγμα, έχει προταθεί ένα σχήμα διαμόρφωσης Slope-Shift Keying LoRa που προσθέτει, πέρα από το κανονικό up-chirp, τη χρήση ενός down-chirp και κυκλικών μετατοπίσεών του για τη μετάδοση επιπλέον πληροφορίας σε κάθε σύμβολο [26]. Με τον τρόπο αυτό αυξάνεται το αλφάριθμο συμβόλων και μπορεί να επιτευχθεί υψηλότερος ρυθμός bit στην ίδια ζώνη συχνοτήτων, παραμένοντας συμβατό με τους δέκτες (απαιτεί όμως νέους αλγόριθμους ανίχνευσης συμβόλων). Μια συναφής προσέγγιση αξιοποιεί τεχνική Index Modulation στα chirps σήματα (ενσωματώνοντας πληροφορία στην επιλογή συγκεκριμένων θέσεων συχνότητας), ώστε να βελτιώσει περαιτέρω τον ρυθμό χωρίς επιπλέον ισχύ ή εύρος ζώνης [27]. Επίσης, έχει παρουσιαστεί μια επέκταση CSS όπου χρησιμοποιούνται ορθογώνια chirps ταυτόχρονα στους τετραγωνικούς άξονες I και Q (δηλ. μετάδοση στο σύμπλοκο επίπεδο με διαμόρφωση IQ) [28]. Αυτή η μέθοδος, γνωστή ως IQ-CSS, γεγονός που αντιστοιχεί θεωρητικά σε μείωση του διαθέσιμου αριθμού bit διπλασιάζει τον ρυθμό bit για το ίδιο BW και SF (καθώς μεταφέρονται διαφορετικά bits στο I και στο Q κανάλι ανά σύμβολο) [28]. Όλες αυτές οι τεχνικές βρίσκονται υπό μελέτη και υπόσχονται σημαντική αύξηση της απόδοσης των LoRa συστημάτων, διατηρώντας παράλληλα τα πλεονεκτήματα μεγάλης εμβέλειας της CSS διαμόρφωσης.

### 2.3.6 Υπολογισμός χρόνου εκπομπής (Time-on-Air) και παράγοντες που τον επηρεάζουν

Ο Χρόνος στον Αέρα ενός πακέτου LoRa, γνωστός ως Time-on-Air (ToA), είναι το χρονικό διάστημα κατά το οποίο το πακέτο εκπέμπεται και καταλαμβάνει το κανάλι. Περιλαμβάνει τόσο τον χρόνο εκπομπής του προοιμίου (preamble) όσο και τον χρόνο εκπομπής του κύριου τμήματος (header + payload). Ο ToA εξαρτάται από μια σειρά παραμέτρων του φυσικού επιπέδου: τον spreading factor, το εύρος ζώνης, το μέγεθος του payload (bytes), τον κωδικό διόρθωσης λαθών CR, την παρουσία ή όχι ρητής επικεφαλίδας (explicit header vs implicit), την ενεργοποίηση του CRC, καθώς και τη ρύθμιση Low Data Rate Optimization (LDRO) που αναφέρθηκε προηγουμένως [12]. Η ακριβής διάρκειά του μπορεί να υπολογιστεί με βάση τις παραμέτρους αυτές, χρησιμοποιώντας τις σχέσεις που ακολουθούν.

Αρχικά, υπενθυμίζεται η διάρκεια συμβόλου  $T_s = 2^{SF}/BW$ . Ένα πακέτο LoRaWAN τυπικά χρησιμοποιεί προοίμιο μήκους  $n_{preamble} = 8$  συμβόλων (που αποτελείται από  $n_{preamble} - 1 = 7$  up-chirps συνένα ειδικό τελικό up-chirp και 2.25 down-chirps για συγχρονισμό στον δέκτη). Επιπλέον, το πρωτόκολλο ορίζει ότι ο δέκτης θα αναζητήσει το προοίμιο με περιθώριο περίπου +4 σύμβολα και μια ίση προκαθορισμένη συσχέτιση = 0.25 συμβόλου για τον συγχρονισμό [12]. Συνολικά, αυτό προσθέτει 4.25 σύμβολα επιπλέον του  $n_{preamble}$  στον υπολογισμό του χρόνου προοιμίου. Επομένως, ο χρόνος προοιμίου είναι:

$$T_{preamble} = (n_{preamble} + 4.25) \cdot T_s \quad (2.15)$$

με  $n_{preamble} = 8$ , έχουμε  $T_{preamble} = 12.25 T_s$ . Για παράδειγμα, αν  $T_s = 1.024ms$  (π.χ.

SF7, 125kHz), το προοίμιο διαρκεί περίπου 12.544ms.

Στη συνέχεια, πρέπει να υπολογιστεί ο αριθμός συμβόλων του payload (μαζί με τυχόν header) που θα μεταδοθούν, δηλαδή πόσα LoRa symbols απαιτούνται για να χωρέσουν τα δεδομένα και η επικεφαλίδα με το επιλεγμένο SF και CR. Η εξίσωση που δίνει τον αριθμό συμβόλων payload ( $N_{payload}$ ) είναι αρκετά σύνθετη και περιλαμβάνει έναν τελεστή ceil (στρογγυλοποίηση προς τα πάνω), καθώς αν δεν προκύπτει ακριβής ακέραιος αριθμός bits από  $N$  σύμβολα, θα απαιτηθεί ένα επιπλέον σύμβολο. Συνοπτικά, η σχέση (όπως προκύπτει από τη Semtech και τα πρότυπα LoRaWAN) είναι η ακόλουθη [11]:

$$N_{payload} = 8 + \max\left\{\left\lceil \frac{8PL - 4SF + 28 + 16CRC - 20H}{4(SF - 2LDRO)} \times (CR + 4) \right\rceil, 0 \right\} \quad (2.16)$$

όπου:

- $PL$  είναι το μέγεθος του payload σε bytes (ωφέλιμα δεδομένα),
- $CRC = 1$  αν συμπεριλαμβάνεται 2 byte CRC στο πακέτο (συνήθως ενεργοποιημένο, αλλιώς 0),
- $H$  είναι bit ένδειξης header:  $H = 0$  για explicit header (παρουσία τυπικής κεφαλίδας LoRaWAN, μήκους 20 bits) ή  $H = 1$  για implicit mode (χωρίς καθόλου header, δηλαδή ο δέκτης γνωρίζει εκ των προτέρων το μέγεθος και το  $CR$ ),
- $LDRO = 1$  αν έχει ενεργοποιηθεί Low Data Rate Optimization (σε  $SF = 11, 12, 125kHz$ , αλλιώς  $LDRO = 0$ )
- $CR$  είναι ο ακέραιος (1-4) δείκτης του rate (όπου ο ρυθμός διόρθωσης είναι  $4/(4+CR)$ ).

Η παραπάνω σχέση μπορεί να αναλυθεί ως εξής: ο αριθμός των bits του αδιαμόρφωτου payload είναι  $8PL$ . Σε αυτό προστίθενται 28 bits επικεφαλίδων πρωτοκόλλου (υπογραμμίζεται ότι όλα τα πακέτα LoRaWAN έχουν μια σταθερή προσθήκη 13 bytes = 26 bits υψηπέδων + 2 bits διαχωρισμού = 28 bits) και δυνητικά 16 bits CRC, ενώ αφαιρούνται 20 bits αν δεν υπάρχει header ( $H = 1$  δηλαδή implicit mode). Η έκφραση  $4(SF - 2LDRO)$  στον παρονομαστή απορρέει από τον τρόπο με τον οποίο ο κώδικας διαμοιράζεται τα bits σε ομάδες των 4 συμβόλων κατά τη διαμόρφωση (interleaving σε τετράδες) και ότι αν το  $LDRO = 1$  μειώνει το αποτελεσματικό SF κατά 2 όπως προαναφέρθηκε. Τέλος, πολλαπλασιάζουμε με  $(CR + 4)$  διότι για κάθε ομάδα 4 payload bits προστίθενται  $CR$  bits FEC (σύμφωνα με τον λόγο  $4/(4+CR)$ ). Ολόκληρο το κλάσμα μέσα στις  $\lceil \rceil$  μας δίνει τον αριθμό τετράδων συμβόλων που απαιτούνται για να χωρέσουν όλα τα bits και οι  $\lceil \rceil$  στρογγυλοποιούν προς τα πάνω στον επόμενο ακέραιο αν υπάρχει υπόλοιπο. Το  $\max\{..., 0\}$  εξασφαλίζει ότι λαμβάνουμε τουλάχιστον 0 (το αποτέλεσμα του κλάσματος μπορεί να βγει αρνητικό για πολύ μικρά πακέτα με συγκεκριμένες παραμέτρους, οπότε θεωρείται 0) [11]. Στο αποτέλεσμα προστίθεται σταθερά το 8, που αντιπροσωπεύει έναν ελάχιστο αριθμό 8 συμβόλων πάντα, ακόμη και για πολύ μικρό payload (πρακτικά το μικρότερο payload που μεταδίδεται καταλαμβάνει 8 σύμβολα, πέραν του preamble).

Αφότου υπολογιστεί το  $N_{payload}$ , μπορούμε να βρούμε τον χρόνο εκπομπής του payload:

$$T_{payload} = N_{payload} T_s \quad (2.17)$$

Τέλος, ο συνολικός χρόνος στον αέρα του πακέτου είναι απλώς το άθροισμα:

$$T_{packet} = T_{preamble} + T_{payload} \quad (2.18)$$

Από τις παραπάνω σχέσεις προκύπτει ότι όσο αυξάνεται ο SF ή/και ο CR, απαιτούνται περισσότερα σύμβολα (μεγαλύτερο  $N_{payload}$ ) και κατά συνέπεια αυξάνεται και το  $T_s$ , οδηγώντας σε μεγαλύτερο χρόνο εκπομπής. Αντίθετα, ένα μεγαλύτερο  $BW$  μειώνει ανάλογα το  $T_s$  (π.χ. διπλασιασμός του  $BW$  μισός χρόνος συμβόλου) και έτσι μειώνει τον ToA. Για να αποδοθεί μια πιο σαφής εικόνα, στον πίνακα 2.4 φαίνεται ο ToA (ms) για ένα σύντομο πακέτο τυπικών διαστάσεων, υπό διάφορους SF, με σταθερά  $BW = 125kHz$  και  $CR = 1(4/5)$ , σύμφωνα με τους υπολογισμούς των παραπάνω εξισώσεων.

Spreading Factor	Symbol Duration (ms)	Low Data Rate Opt.
7	41	
8	72	
9	144	
10	289	
11	578	(LDRO = 1)
12	991	(LDRO = 1)

Πίνακας 2.4: Διάρκεια συμβόλου LoRa σε  $BW=125 kHz$  για διάφορα Spreading Factors, με ενεργοποιημένη Low Data Rate Optimization (LDRO) όπου απαντείται.

Οι τιμές αυτές επιβεβαιώνουν ότι περίπου κάθε μονάδα αύξησης του SF διπλασιάζει τον απαιτούμενο χρόνο εκπομπής (για σταθερό εύρος ζώνης) [29]. Αυτό έχει σημαντικές επιπτώσεις στην ενεργειακή κατανάλωση των κόμβων: ένας κόμβος που εκπέμπει με υψηλό SF θα διατηρεί τον πομπό του ανοικτό πολύ περισσότερη ώρα σε σχέση με έναν που εκπέμπει το ίδιο δεδομένο με χαμηλότερο SF, ξοδεύοντας περισσότερη ενέργεια από την μπαταρία του. Για τον λόγο αυτό, η επιλογή του χαμηλότερου δυνατού SF που ικανοποιεί τις απαιτήσεις επικοινωνίας είναι κρίσιμη για την παράταση της διάρκειας ζωής των συσκευών. Επιπλέον, σε κανονιστικά πλαίσια όπως η Ευρώπη, ισχύουν περιορισμοί duty cycle στο EU863-870 που διαφέρουν ανά υπο-ζώνη (π.χ. 0.1%, 1% ή 10%). Όσο μεγαλώνει ο ToA, τόσο περιορίζεται ο πρακτικός ρυθμός αποστολών. Ενδεικτικά, αν ένα πακέτο διαρκεί 1 δευτερόλεπτο στον αέρα και το όριο της υπο-ζώνης είναι 1%, η συσκευή δεν μπορεί να το εκπέμπει πάνω από 36 φορές (περίπου) ανά ώρα. Συνεπώς, η μείωση ToA βοηθά ταυτόχρονα την ενεργειακή απόδοση και τη συμμόρφωση με τους περιορισμούς εκπομπής.

### 2.3.7 Αποδιαμόρφωση και αποκωδικοποίηση σήματος LoRa

Η λήψη και αποδιαμόρφωση (demodulation) του σήματος LoRa πραγματοποιείται με τεχνικές που εκμεταλλεύονται τη δομή των chirp. Ο δέκτης, μόλις ανιχνεύσει το προοίμιο, παράγει ένα τοπικό σήμα αναφοράς (π.χ. έναν down-chirp που καλύπτει την ίδια ζώνη συ-

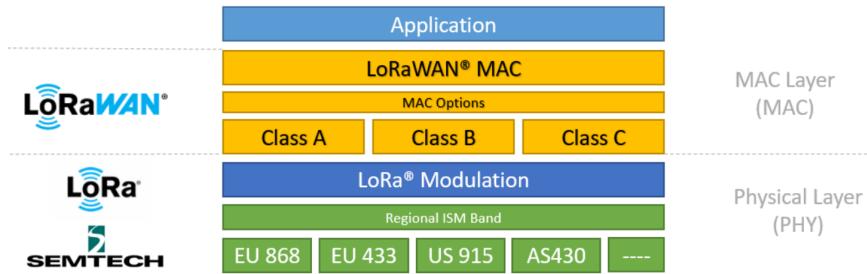
χνοτήτων) και πολλαπλασιάζει το ληφθέν σήμα με το συζυγές του σήματος αναφοράς, μία διαδικασία γνωστή ως «απο-τσιρπικοποίηση» (de-chirping). Με αυτόν τον τρόπο, ένας λαμβανόμενος up-chirp μετατρέπεται σε ένα σχεδόν σταθερής συχνότητας ημιτονικό σήμα στο πεδίο του χρόνου, του οποίου η συχνότητα εξαρτάται από τη σχετική μετατόπιση (ολίσθηση) που είχε το chirp του συμβόλου. Εφαρμόζοντας έναν γρήγορο μετασχηματισμό Fourier (FFT) στο απο-διαμόρφωμένο σήμα, ο δέκτης μπορεί να αποτυπώσει ένα διάγραμμα ισχύος ως προς τη συχνότητα, στο οποίο εμφανίζεται μια χαρακτηριστική κορυφή. Η θέση της κορυφής αυτής στο φάσμα αντιστοιχεί στη δυαδική τιμή του συμβόλου που μεταδόθηκε. Με άλλα λόγια, η αποδιαμόρφωση στο LoRa υλοποιείται με έναν αποδιασπορέα συχνότητας και έναν μετασχηματισμό FFT, που επιτρέπουν την ανίχνευση συσχέτισης του σήματος με κάθε πιθανή συμβολοσειρά [11, 28, 21]. Η μέθοδος αυτή είναι αποδοτική υπολογιστικά και μπορεί να υλοποιηθεί με χαμηλής κατανάλωσης ηλεκτρονικά εξαρτήματα στον δέκτη, σε αντίθεση με πιο περίπλοκα σχήματα διάχυσης (π.χ. DSSS).

Αφότου εξαχθούν οι διαδοχικές τιμές συμβόλων από το φυσικό επίπεδο, ακολουθεί η διαδικασία αποκωδικοποίησης των bits. Σε φυσικό επίπεδο, το LoRa εφαρμόζει διορθωτικό κώδικα προς διόρθωση σφαλμάτων (Forward Error Correction). Συγκεκριμένα, χρησιμοποιείται ένας κώδικας διεύρυνσης (coding rate) όπου για κάθε ομάδα 4 δεδομένων bits προστίθενται CR επιπλέον bits ανίχνευσης/διόρθωσης σφαλμάτων. Αυτό αντιστοιχεί σε λόγο κώδικα  $4/(4+CR)$ . Για παράδειγμα  $CR = 1$  δίνει  $4/5$  (20% πλεονάζοντα bits), ενώ  $CR = 4$  δίνει  $4/8=1/2$  (50% πλεονάζοντα bits). Ο δέκτης, γνωρίζοντας το CR από την επικεφαλίδα (ή προκαθορισμένα), εφαρμόζει τον αντίστροφο αλγόριθμο του κώδικα για να ανιχνεύσει και να διορθώσει τυχόν σφάλματα στα bits των συμβόλων. Έπειτα, ελέγχει την ακεραιότητα της συνολικής ωφέλιμης ακολουθίας μέσω του CRC (εάν υπάρχει). Εφόσον το CRC επαληθευτεί, τα αρχικά δεδομένα περνούν στο επάνω επίπεδο (π.χ. εφαρμογή). Σημειώνεται ότι η ανοχή του LoRa σε ταυτόχρονες μεταδόσεις περιορίζεται όταν αυτές χρησιμοποιούν ίδιο SF και κανάλι. Σε τέτοιες περιπτώσεις (collision), ο δέκτης συνήθως θα συγχρονιστεί και θα αποδιαμόρφωσει μόνο το ισχυρότερο από τα πακέτα (φαινόμενο capture effect), εκτός αν τα σήματα είναι επαρκώς χρονικά και φασματικά διαχωρισμένα. Παρ' όλα αυτά, η χρήση διαφορετικών Spreading Factors ή καναλιών συχνοτήτων από τις συσκευές αποτρέπει τις περισσότερες συγκρούσεις, αξιοποιώντας πλήρως την ορθογωνιότητα και ευελιξία του πρωτοκόλλου.

## 2.4 Το πρωτόκολλο LoRaWAN

Μετά την ανάλυση της φυσικής στρώσης LoRa και των δικτύων LPWAN στις προηγούμενες ενότητες, στρεφόμαστε τώρα στο πρωτόκολλο LoRaWAN. Το LoRaWAN είναι ένα ανοικτό πρωτόκολλο δικτύου που αναπτύσσεται από τη συμμαχία LoRa Alliance και λειτουργεί πάνω από τη διαμόρφωση LoRa, καθορίζοντας πώς οι συσκευές επικοινωνούν σε επίπεδο MAC και δικτύου. Ενώ το LoRa εξασφαλίζει τη μετάδοση σε μεγάλες αποστάσεις με χαμηλή ισχύ, το LoRaWAN ορίζει την αρχιτεκτονική και τους κανόνες δικτύωσης ώστε εκατομμύρια τερματικές συσκευές να μπορούν να συνδεθούν αξιόπιστα μέσω μίας υποδομής μεγάλης κλίμακας. Το πρωτόκολλο αυτό έχει σχεδιαστεί ειδικά για τις απαιτήσεις του IoT: αμφίδρομη επικοινωνία με χαμηλό ρυθμό δεδομένων, ασφάλεια από άκρο σε άκρο, δυνατότητα κινητικότητας και στήριξη υπηρεσιών εντοπισμού θέσης [30]. Παρακάτω παρουσιάζονται αναλυτικά η αρχι-

τεκτονική του LoRaWAN, οι κατηγορίες λειτουργίας των συσκευών, οι τύποι μηνυμάτων και η μορφή του πλαισίου, οι μηχανισμοί ενεργοποίησης και ασφάλειας, καθώς και σημαντικές παραμέτροι όπως το Adaptive Data Rate και οι περιορισμοί εκπομπής, κάνοντας αναφορές στα επίσημα πρότυπα (LoRa Alliance) και πρόσφατη βιβλιογραφία όπου ενδείκνυται. Τέλος, παρουσιάζονται συνοπτικά στοιχεία του οικοσυστήματος LoRaWAN στην πράξη (π.χ. The Things Network) και συνδέσεις με εφαρμογές όπως η τηλεμέτρηση ενέργειας με έξυπνους μετρητές σε υποσταθμούς ηλεκτρικής ενέργειας.



Εικόνα 2.9: Τεχνολογική στοιβα των LoRa και LoRaWAN.

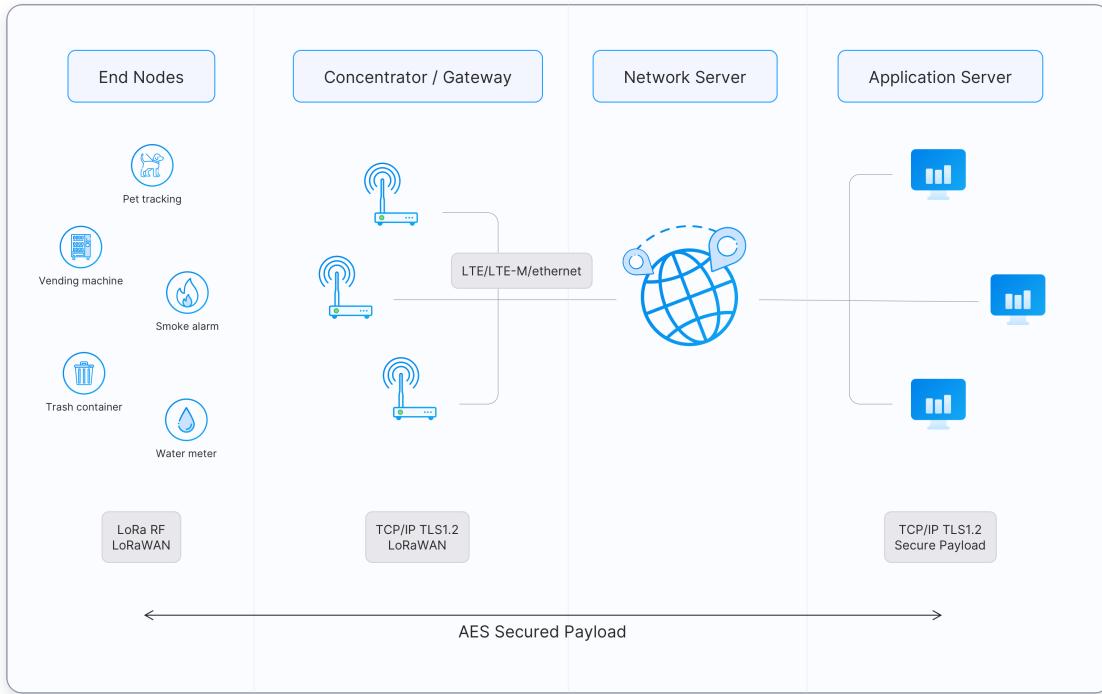
[7]

#### 2.4.1 Αρχιτεκτονική δικτύου LoRaWAN

Το δίκτυο LoRaWAN υλοποιείται σε τοπολογία τύπου «star-of-stars» (αστέρι των αστεριών), όπου οι πύλες (gateways) λειτουργούν ως διαμεσολαβητές μεταφέροντας ασύρματα μηνύματα μεταξύ των τερματικών συσκευών και ενός κεντρικού Network Server. Στην Εικόνα 2.10 απεικονίζεται μια τυπική αρχιτεκτονική LoRaWAN με τις τερματικές συσκευές (end devices) να επικοινωνούν μέσω πολλαπλών πυλών με έναν κεντρικό Network Server, ενώ τα δεδομένα προωθούνται τελικά σε έναν Application Server, όπου βρίσκεται η τελική εφαρμογή του χρήστη. Το LoRaWAN Network Server (LNS) είναι υπεύθυνο για τον έλεγχο και συντονισμό του δικτύου, ενώ ένας ξεχωριστός Join Server μπορεί να συμμετέχει στη διαδικασία εισόδου νέων συσκευών στο δίκτυο, αποθηκεύοντας τα απαραίτητα κλειδιά και συνδράμοντας στον υπολογισμό των κλειδιών ασφαλείας κατά την ενεργοποίηση (περισσότερα για αυτό στην Υποενότητα 2.4.4) [16].

**Τερματικές Συσκευές (End Devices):** Πρόκειται για αισθητήρες, μετρητές ή ενεργοποιητές που διαθέτουν πομποδέκτη LoRa. Είναι συνήθως συσκευές χαμηλής ισχύος, συχνά με μπαταρία, που στέλνουν δεδομένα (uplinks) ή λαμβάνουν εντολές (downlinks) ασύρματα. Κάθε τερματική συσκευή επικοινωνεί απευθείας με όποιες πύλες βρίσκονται στην εμβέλειά της, χρησιμοποιώντας την ασύρματη ζεύξη LoRa χωρίς ανάγκη συσχέτισης με συγκεκριμένη πύλη. Η μετάδοση είναι τύπου ALOHA, δηλαδή χωρίς χειραφία και μπορεί να ληφθεί από πολλές πύλες ταυτόχρονα [16]. Η κάθε συσκευή αναγνωρίζεται στο δίκτυο από μια διεύθυνση συσκευής (DevAddr) μήκους 32 bit, που εκχωρείται κατά την ενεργοποίηση.

**Πύλες (Gateways):** Οι πύλες λειτουργούν ως διαφανείς γέφυρες που μετατρέπουν τα ασύρματα πακέτα LoRa σε πακέτα IP και αντιστρέφονται [30]. Μια πύλη LoRaWAN περιλαμβάνει δέκτη και πομπό LoRa (συχνά με δυνατότητα ταυτόχρονης λήψης σε πολλαπλά κανάλια συχνοτήτων) και συνδέεται μέσω δικτύου IP (Ethernet, Wi-Fi, ή κινητή σύνδεση



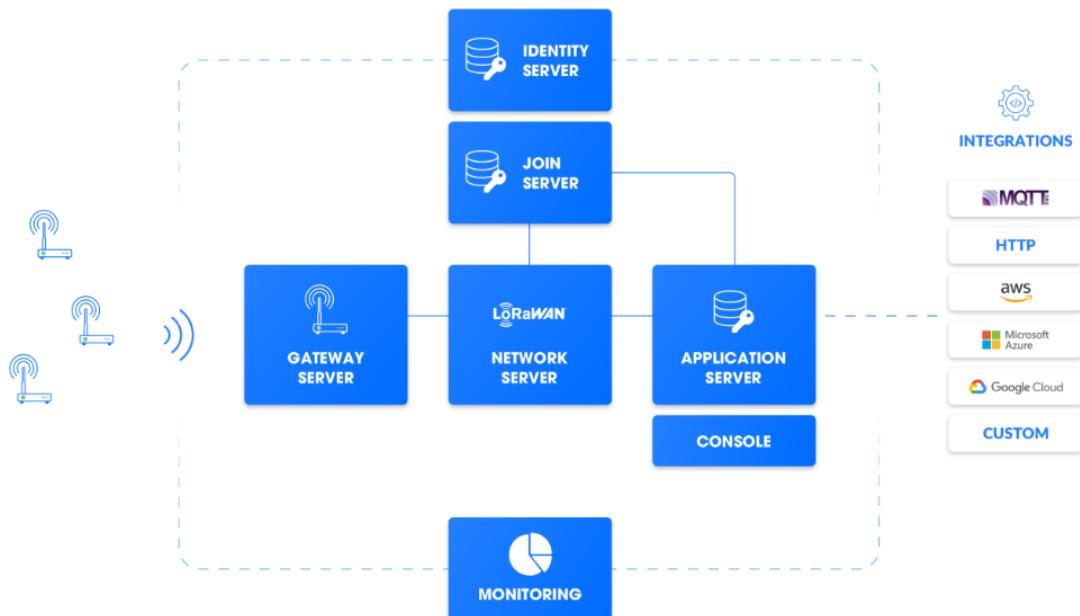
Εικόνα 2.10: Τυπική αρχιτεκτονική LoRaWAN δικτύου.

[16]

LTE/5G) με τον Network Server. Δεν πραγματοποιεί τοπική αναμετάδοση ή δρομολόγηση, αντίθετα κάθε λαμβανόμενο πλαίσιο LoRa προωθείται αυτούσιο στον Network Server. Σε αντίθεση με τα δίκτυα κινητής, οι πύλες LoRaWAN δεν διαχειρίζονται συσχετίσεις σύνδεσης. Οποιαδήποτε πύλη που λαμβάνει ένα έγκυρο πακέτο από μια συσκευή θα το μεταδώσει προς το κεντρικό δίκτυο. Αυτό επιτρέπει πλεονασμό και ευρεία κάλυψη, καθώς ένα uplink μήνυμα μπορεί να ληφθεί από πολλές πύλες παράλληλα. Ο Network Server φροντίζει να απορρίψει τα διπλότυπα και να κρατήσει μόνο ένα αντίγραφο (συνήθως από την πύλη με το καλύτερο σήμα). Οι πύλες αποτελούν το μοναδικό σημείο εκπομπής downlink μηνυμάτων από το δίκτυο προς τις συσκευές [16]. Αξίζει να σημειωθεί ότι σε εφαρμογές διαχείρισης δικτύων ενέργειας ή βιομηχανικών εγκαταστάσεων, οι πύλες LoRaWAN μπορούν να τοποθετηθούν π.χ. σε υποσταθμούς ή κέντρα ελέγχου ώστε να συλλέγουν δεδομένα από πολλούς κατανεμημένους αισθητήρες (για παράδειγμα, μετρητές κατανάλωσης) στην περιοχή.

**Network Server (Διακομιστής Δικτύου):** Ο Network Server (NS) είναι η «καρδιά» του δικτύου LoRaWAN. Πρόκειται για λογισμικό που τρέχει σε κεντρικό διακομιστή (cloud ή on-premise) και επιτελεί μια σειρά από κρίσιμες λειτουργίες δικτύου: i) Επικυρώνει την αυθεντικότητα των συσκευών και την ακεραιότητα των μηνυμάτων, ελέγχοντας τον Message Integrity Code (MIC) κάθε πλαισίου με τα κατάλληλα κλειδιά. ii) Κάνει απαλοιφή διπλοτύπων (deduplication) για uplink πακέτα που έλαβε ταυτόχρονα από πολλαπλές πύλες. iii) Καταχωρεί και διαχειρίζεται τις ενεργές συσκευές και τις διευθύνσεις τους (DevAddr), ελέγχοντας επίσης το εύρος των frame counters για αποτροπή επαναλήψεων (replay attacks). iv) Δρομολογεί τα εξερχόμενα application payloads προς τους αντίστοιχους Application Servers και αντίστροφα, λαμβάνοντας από αυτούς καθοδηγούμενα downlink μηνύματα για τις συσκευές. v) Επιλέγει την πλέον κατάλληλη πύλη για να μεταδώσει ένα downlink προς μια συσκευή

(συνήθως την πύλη που είχε το καλύτερο σήμα στο τελευταίο uplink του εν λόγω κόμβου). vi) Αποστέλλει εντολές διαχείρισης σύνδεσης και πόρων, όπως τις εντολές ADR (Adaptive Data Rate) προς τις συσκευές, ρυθμίζοντας τον ρυθμό δεδομένων ή την ισχύ εκπομπής τους για βελτιστοποίηση της ενεργειακής κατανάλωσης και της χωρητικότητας του δικτύου. vii) Συντονίζει τις διαδικασίες ενεργοποίησης συσκευών (OTAA), προωθώντας τα σχετικά μηνύματα Join προς τον κατάλληλο Join Server και διασφαλίζοντας την ορθή διανομή των κλειδιών ασφαλείας (βλ. Υπενότητα 2.4.4). Συνολικά, ο Network Server υλοποιεί ολόκληρο το πρωτόκολλο LoRaWAN στη μεριά του δικτύου και ενεργεί ως το μόνο σημείο λήψης αποφάσεων για τη ροή των δεδομένων και τον έλεγχο των συσκευών [16].



Εικόνα 2.11: Αρχιτεκτονική LoRaWAN Network Server.  
[31]

**Application Server (Διακομιστής Εφαρμογών):** Ο Application Server (AS) είναι υπεύθυνος για την παραλαβή και επεξεργασία των δεδομένων εφαρμογής από τις συσκευές, καθώς και για τη δημιουργία τυχόν downlink μηνυμάτων σε επίπεδο εφαρμογής. Στο LoRaWAN η ασφάλεια είναι διαμοιρασμένη, έτσι ο Application Server διατηρεί το κλειδί εφαρμογής (AppSKey) για κάθε συσκευή, προκειμένου να αποκρυπτογραφεί τα δεδομένα που προωθεί ο Network Server. Οποιαδήποτε επιχειρησιακή λογική (π.χ. αποθήκευση μετρήσεων, ανάλυση δεδομένων, εμφάνιση σε πίνακες ελέγχου) υλοποιείται πάνω από τον Application Server. Σημειώνεται ότι μπορεί να υπάρχουν πολλαπλοί Application Servers σε ένα δίκτυο (π.χ. διαφορετικοί οργανισμοί να λαμβάνουν δεδομένα από τις δικές τους συσκευές) και ο Network Server φροντίζει να δρομολογεί σωστά τα πακέτα σε καθέναν από αυτούς (βάσει του DevAddr/εφαρμογής που αντιστοιχεί στη συσκευή) [16].

**Join Server:** Ο Join Server είναι μια επιπρόσθετη οντότητα (διακομιστής) που εμφανίστηκε κυρίως μετά την έκδοση LoRaWAN 1.1. και αναλαμβάνει να διαχειρίζεται την ασφαλή ενεργοποίηση των συσκευών. Συγκεκριμένα, ο Join Server αποθηκεύει τα Root Keys των συσκευών (βασικά κλειδιά εγγραφής, όπως το AppKey/NwkKey) και συμμετέχει

στη διαδικασία Over-The-Air Activation (OTAA) υπολογίζοντας και παρέχοντας τα προσωρινά κλειδιά συνεδρίας τόσο στον Network Server όσο και στον Application Server [32]. Με αυτόν τον διαχωρισμό, επιτυγχάνεται απομόνωση των πεδίων ασφαλείας: ο Network Server δεν χρειάζεται να γνωρίζει το κλειδί εφαρμογής της συσκευής, ενώ ο Application Server δεν γνωρίζει τα κλειδιά δικτύου, ενώ αμφότεροι λαμβάνουν μόνο τα κλειδιά συνεδρίας που τους αναλογούν από τον Join Server. Στις πρώτες εκδόσεις (1.0.x) ο ρόλος του Join Server είτε δεν υπήρχε (το AppKey ήταν γνωστό απευθείας στο Network Server) είτε μπορούσε να συμπίπτει με τον Application Server. Στο LoRaWAN 1.1 όμως καθορίζεται ρητά ξεχωριστός Join Server, βελτιώνοντας την ασφάλεια και υποστηρίζοντας επιπλέον λειτουργίες όπως η περιαγωγή μεταξύ δικτύων [16, 33]. Η διαδικασία OTAA με τη συμμετοχή Join Server περιγράφεται λεπτομερώς στην Υποενότητα 2.4.4.

Τέλος, αξίζει να αναφέρουμε ότι η αρχιτεκτονική end-to-end του LoRaWAN δεν οριοθετεί αυστηρά το εμπορικό μοντέλο υλοποίησης, μιας και μπορούν να υπάρξουν δημόσια δίκτυα, ιδιωτικά δίκτυα ή κοινόχρηστες υποδομές. Το πρότυπο εγγυάται τη διαλειτουργικότητα, δηλαδή μια πιστοποιημένη συσκευή LoRaWAN μπορεί να λειτουργήσει σε οποιοδήποτε συμβατό δίκτυο. Για παράδειγμα, το The Things Network (TTN) αποτελεί ένα παγκόσμιο δημόσιο/community δίκτυο LoRaWAN, ενώ υπάρχουν πάροχοι που προσφέρουν εμπορικές υποδομές (Orange, LORIOT κ.ά.), καθώς και δυνατότητα για εντελώς ιδιωτικά δίκτυα (π.χ. εγκατάσταση ενός Network Server και gateways αποκλειστικά για μια βιομηχανική εγκατάσταση ή ένα δίκτυο ενέργειας).

## 2.4.2 Κλάσεις συσκευών και χρονισμοί επικοινωνίας

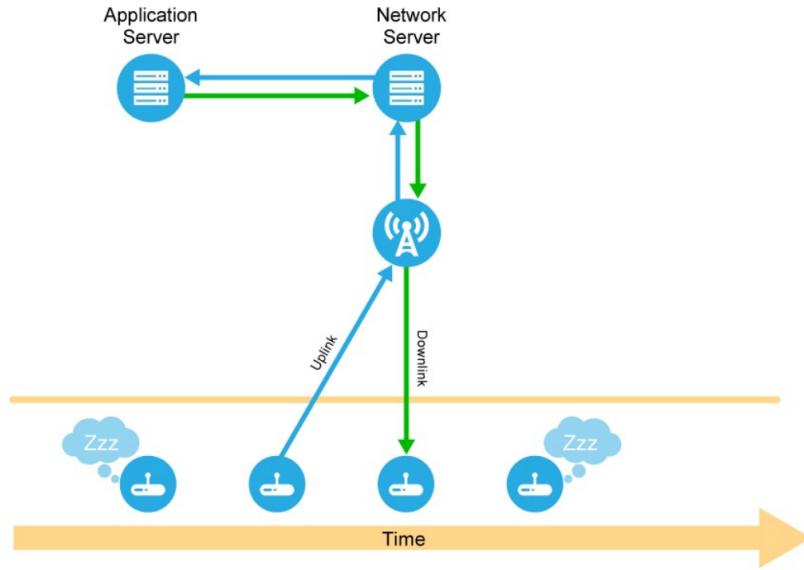
Το LoRaWAN υποστηρίζει τρεις κλάσεις λειτουργίας τερματικών συσκευών, τις Class A, Class B και Class C, ώστε να εξυπηρετεί διαφορετικές απαιτήσεις εφαρμογών σε όρους κατανάλωσης ισχύος, καθυστέρησης (latency) των downlink μηνυμάτων και λειτουργικότητας. Όλες οι συσκευές οφείλουν να υποστηρίζουν τουλάχιστον την Κλάση A (βασική λειτουργία), ενώ οι B και C είναι προαιρετικές επεκτάσεις. Η επικοινωνία σε όλες τις κλάσεις είναι αμφίδρομη (υπάρχει δυνατότητα για uplink και downlink), με διαφορετικά όμως χρονικά μοτίβα για τα παράθυρα μετάδοσης και λήψης ανά κλάση. Ακολουθεί περιγραφή κάθε κλάσης και των χρονισμών της, με αναφορά στους καθορισμένους χρόνους παραθύρων λήψης.

Αξίζει να σημειωθεί ότι, κατά τη διάρκεια της μετάδοσης ενός uplink, δεν μπορούν οι συσκευές να λάβουν downlink. Η λήψη τους πραγματοποιείται μόνο στα καθορισμένα παράθυρα, εκτός αν η συσκευή ανήκει στην Class C και βρίσκεται σε συνεχή λήψη (listen mode).

### 2.4.2.1 Class A - Υποχρεωτική (βασική) κλάση, ελάχιστης ισχύος

Στην Κλάση A ανήκουν εξ ορισμού όλες οι συσκευές LoRaWAN. Πρόκειται για το πιο αποδοτικό ενεργειακά «modus operandi»,<sup>3</sup> όπου η συσκευή μεταβαίνει στο δίκτυο μόνο όταν έχει δεδομένα να στείλει. Κάθε συσκευή Class A μπορεί να εκπέμψει ένα uplink ανά πάσα στιγμή, αναλόγως την εφαρμογή της (asynchronous uplink), ξεκινώντας έτσι την

<sup>3</sup>Η λατινική φράση «**modus operandi**» αναφέρεται στον χαρακτηριστικό τρόπο λειτουργίας ή στη μεθοδολογία εφαρμογής μιας διαδικασίας.



Εικόνα 2.12: Ροή Uplink και Downlink μηνυμάτων στο Πρωτόκολλο LoRaWAN.

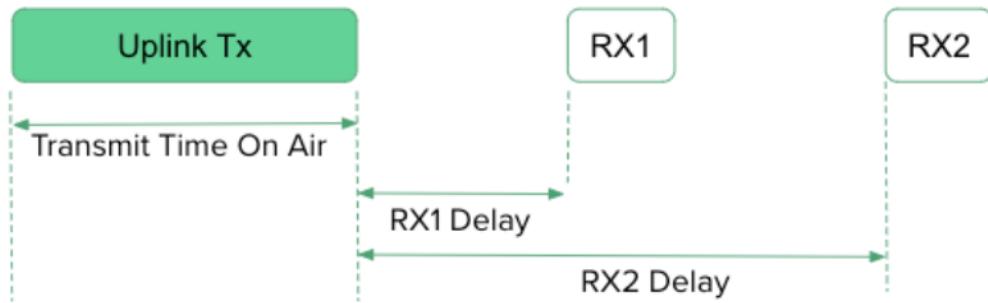
[34]

επικοινωνία. Αφότου στείλει ένα uplink, η συσκευή ανοίγει δύο σύντομα παράθυρα λήψης αναμένοντας κάποιο downlink από τον διακομιστή (Network Server). Το πρώτο παράθυρο (RX1) ξεκινά, τυπικά, περίπου 1 δευτερόλεπτο μετά το τέλος της μετάδοσης, ενώ το δεύτερο (RX2) ακολουθεί περίπου 1 δευτερόλεπτο αργότερα από το πρώτο, σύμφωνα με τις προδιαγραφές του πρωτοκόλλου (βλ. Εικόνα 2.13). Αν ο Network Server έχει έτοιμο ένα downlink για τη συσκευή, μπορεί να το στείλει στο RX1 ή (αν χαθεί το πρώτο) στο RX2. Εάν δεν σταλεί κανένα μήνυμα σε κανένα παράθυρο, η συσκευή επιστρέφει σε κατάσταση «ύπνου» μέχρι το επόμενο uplink. Να σημειωθεί ότι τα παράθυρα RX1/RX2 είναι πολύ σύντομα (της τάξης μερικών δεκαδών ή εκατοντάδων ms) και προκαθορισμένα, ώστε να ελαχιστοποιείται η ενεργοβόρος κατάσταση λήψης του ραδιοφώνου. Η Class A ελαχιστοποιεί την κατανάλωση ενέργειας, μιας και η συσκευή κοιμάται το μέγιστο δυνατό διάστημα και «ξυπνά» μόνο για μετάδοση και λήψη. Η καθυστέρηση παράδοσης ενός downlink, όμως, μπορεί να είναι μεγάλη και μη προσδιορίσιμη, καθώς το δίκτυο οφείλει να περιμένει μέχρι το επόμενο uplink της συσκευής για να της στείλει ξανά δεδομένα (αφού μόνο τότε ανοίγουν τα RX παράθυρα).

Συνεπώς η Class A είναι κατάλληλη για εφαρμογές όπου τα downlinks είναι σπάνια ή όχι επείγοντα και όπου η διάρκεια ζωής της μπαταρίας είναι υψηλής σημασίας (επιτυγχάνεται διάρκεια πολλών ετών). Τυπικά παραδείγματα Class A συσκευών είναι οι περιβαλλοντικοί αισθητήρες, οι ανιχνευτές καπνού, οι ιχνηλάτες ζώων ή αντικειμένων κ.ά.

#### 2.4.2.2 Class B - Προγραμματισμένα παράθυρα λήψης με φάρο συγχρονισμού

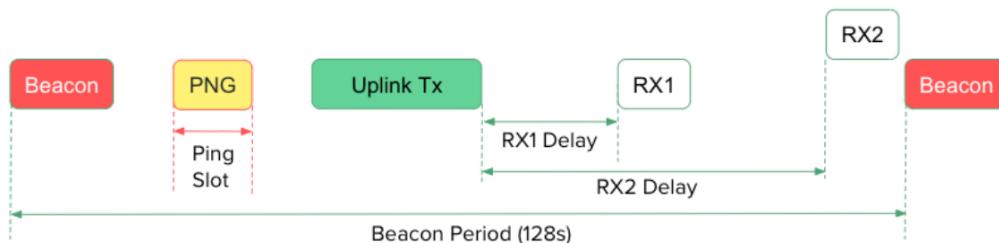
Η Κλάση B επεκτείνει τη συμπεριφορά της κλάσης A, εισάγοντας επιπλέον περιοδικά καθορισμένα παράθυρα λήψης downlink, ανεξάρτητα αν έχει σταλθεί προηγουμένως κάποιο uplink. Πιο συγκεκριμένα, το δίκτυο μεταδίδει ειδικά σήματα συγχρονισμού, τα Beacons (φάροι), σε τακτά χρονικά διαστήματα (π.χ. κάθε 128 δευτερόλεπτα). Οι συσκευές Class B λαμβάνουν αυτούς τους φάρους και συγχρονίζουν ένα εσωτερικό ρολόι. Έτσι γνωρίζουν πότε ακριβώς πρέπει να «αφουγκραστούν» για τυχόν downlink, ανοίγοντας σύντομα παράθυ-



Εικόνα 2.13: Ροή επικοινωνίας κλάσης A.

ρα λήψης, τα ping slots, σε καθορισμένα χρονικά διαστήματα (π.χ. μερικά δευτερόλεπτα ή λεπτά μεταξύ τους). Ο Network Server, γνωρίζοντας το πρόγραμμα των ping slots για κάθε Class B συσκευή (καθώς και ένα «bitmask» χρονικού διαμοιρασμού που δηλώνει πόσο συχνά ανοίγουν τα slots), μπορεί να στείλει downlink με καθορισμένη καθυστέρηση που συμπίπτει με κάποιο από αυτά τα slot. Έτσι εξασφαλίζεται πεπερασμένη και σχετικά μικρή καθυστέρηση παράδοσης downlink (π.χ. μια συσκευή μπορεί να λαμβάνει ως εγγύηση ότι θα ακούει κάθε 32 δευτερόλεπτα, οπότε ο χρόνος απόφασης για downlink είναι το πολύ 32 δευτερόλεπτα). Φυσικά, αυτή η βελτίωση συνεπάγεται ελαφρώς υψηλότερη κατανάλωση, αφού η συσκευή πρέπει να διατηρεί ενεργό τον δέκτη της για τα ping slots και να λαμβάνει τους περιοδικούς φάρους. Παρ' όλα αυτά, το επιπλέον φορτίο είναι σχετικά μικρό (ο χρόνος λήψης είναι σύντομος και ο φάρος μεταδίδεται σε αραιή συχνότητα), επιτρέποντας στις Class B συσκευές να μπορούν να τροφοδοτηθούν μόνο με μπαταρία.

Η Class B είναι κατάλληλη όταν απαιτείται εγγυημένος χρόνος απόκρισης σε κλίμακα δευτερολέπτων ή δεκάδων δευτερολέπτων για downlink μηνύματα. Τυπικό παράδειγμα αποτελούν ασύρματοι μετρητές αφέλιμων υπηρεσιών (ηλεκτρικού, ύδρευσης κ.λπ.), που ανήκουν στην κλάση B, όπου οι κόμβοι παραμένουν κυρίως σε ύπνο αλλά συγχρονίζονται με beacons και ανοίγουν προγραμματισμένα ping slots. Έτσι, εντολές όπως αλλαγές ρυθμίσεων ή firmware updates μπορούν να παραδοθούν μέσα σε προκαθορισμένο χρόνο (ο οποίος καθορίζεται από την περίοδο του beacon και τη συχνότητα των ping slots) και όχι απλώς όταν τύχει το επόμενο uplink όπως στην Class A. Σημειώνεται, επίσης, ότι οι συσκευές Class B διατηρούν πλήρως και τη συμπεριφορά της Class A, δηλαδή κάθε uplink τους ακολουθείται από τα παράθυρα RX1/RX2, ώστε το δίκτυο να μπορεί να απαντήσει άμεσα χωρίς να αναμένει το επόμενο προγραμματισμένο slot.



Εικόνα 2.14: Ροή επικοινωνίας κλάσης B.

### 2.4.2.3 Class C - Συνεχής λήψη (χαμηλή λανθάνουσα, υψηλότερη ισχύς)

Η Κλάση C διευρύνει την Class A προς την αντίθετη κατεύθυνση. Ειδικότερα, οι συσκευές διατηρούνται σχεδόν συνεχώς σε κατάσταση ακρόασης για downlink, εξασφαλίζοντας ελάχιστη καθυστέρηση στην παράδοση εντολών. Συγκεκριμένα, μια συσκευή Class C ανοίγει το δεύτερο παράθυρο λήψης (RX2) διαρκώς, αμέσως μετά το τέλος του σύντομου RX1 και μέχρι την επόμενη μετάδοσή της. Έτσι, πρακτικά, εκτός από τις στιγμές που εκπέμπει η ίδια (όπου προφανώς δεν μπορεί ταυτόχρονα να λάβει), ο δέκτης της παραμένει μονίμως ενεργός. Το αποτέλεσμα είναι ότι ο Network Server μπορεί να στείλει ένα downlink ανά πάσα στιγμή σε μια Class C συσκευή (δεν χρειάζεται να περιμένει uplink ή προκαθορισμένο slot), οπότε η καθυστέρηση είναι μηδενική από την οπτική του προγραμματισμού (περιορίζεται μόνο από τον χρόνο διάδοσης και προετοιμασίας του πακέτου).

Αυτή η λειτουργία ενδείκνυται για εφαρμογές που απαιτούν άμεση αντίδραση ή συνεχή έλεγχο των συσκευών μέσω downlink, π.χ. απομικρυσμένος έλεγχος βιομηχανικού εξοπλισμού, έξυπνος φωτισμός δρόμων (που μπορεί να χρειάζεται εντολές on/off με μικρή καθυστέρηση) ή ακόμη και μετρητές ρεύματος που είναι δικτυωμένοι στο ρεύμα (οπότε δεν έχουν περιορισμό μπαταρίας και μπορούν να ακούν συνεχώς για να λαμβάνουν ενημερώσεις σε πραγματικό χρόνο). Το μειονέκτημα φυσικά είναι η αυξημένη κατανάλωση. Μία Class C συσκευή πρέπει να τροφοδοτεί τον δέκτη της συνεχώς, κάτι που τυπικά καταναλώνει τάξης δεκάδων mW συνεχώς, καθιστώντας την ακατάλληλη για μακροχρόνια λειτουργία με χρήση μπαταρίας. Ως εκ τούτου, σχεδόν όλες οι Class C συσκευές είναι συνδεδεμένες σε μόνιμη παροχή ρεύματος ή χρησιμοποιούνται μόνο περιστασιακά ως Class C (το πρότυπο επιτρέπει δυναμική εναλλαγή κλάσης, π.χ. μια συσκευή μπορεί προσωρινά να περάσει σε Class C όταν έχει πρόσθιαση σε εξωτερική τροφοδοσία για μια ενημέρωση λογισμικού και μετά να επιστρέψει σε Class A). Σχηματικά, όπως δείχνει η Εικόνα 2.15, το RX2 παράθυρο παραμένει ανοιχτό επ' αόριστον μέχρι να χρειαστεί η ίδια η συσκευή να στείλει νέο uplink, οπότε διακόπτει στιγμιαία τη λήψη για να εκπέμψει (half-duplex λειτουργία).



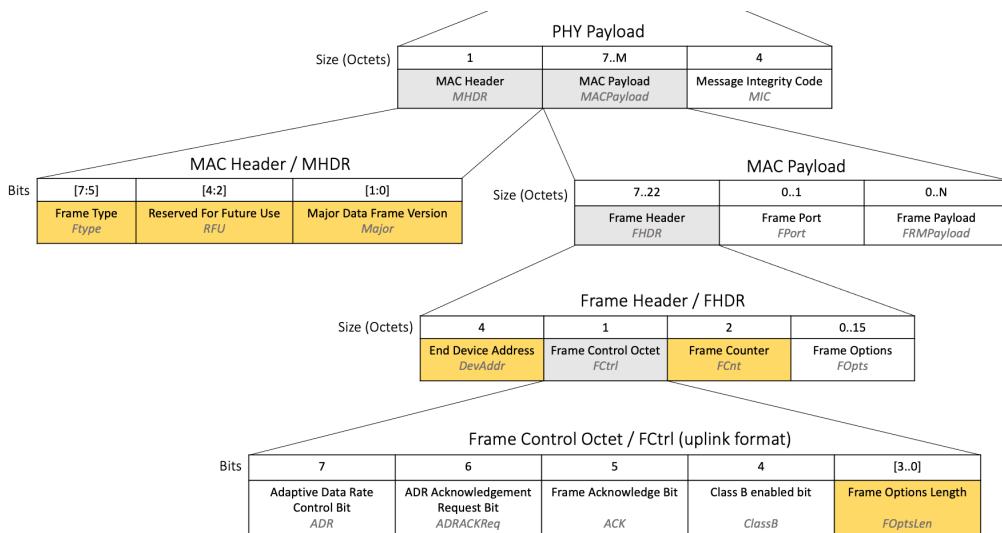
Εικόνα 2.15: Ροή επικοινωνίας κλάσης C.

Συνοψίζοντας, οι τρεις κλάσεις του LoRaWAN προσφέρουν ένα φάσμα επιλογών μεταξύ ελάχιστης ενεργειακής κατανάλωσης (Class A) και ελάχιστου χρόνου απόκρισης (Class C), με έναν ενδιαφέροντα ενδιάμεσο συμβιθασμό, την Class B για εφαρμογές όπου απαιτείται περιοδική επικοινωνία χαμηλής καθυστέρησης. Όλοι οι μηχανισμοί αυτοί υλοποιούνται σε επίπεδο MAC, «διάφανη» προς την εφαρμογή, επιτρέποντας στο ίδιο δίκτυο να εξυπηρετεί ποικίλες συσκευές. Σημειώνεται επίσης ότι το LoRaWAN υποστηρίζει και ομαδική επικοι-

νωνία multicast για downlink σε πολλές συσκευές ταυτόχρονα (π.χ. για μαζική αναβάθμιση firmware). Οι συσκευές μπορούν να οριστούν σε γκρουπς και τα downlinks πολυεκπομπής να παραδίδονται κατά προτίμηση σε Class B ή Class C συσκευές για να εξασφαλιστεί η λήψη τους.

### 2.4.3 Τύποι μηνυμάτων και δομή πλαισίου LoRaWAN

Η επικοινωνία στο LoRaWAN γίνεται μέσω δομών δεδομένων που ονομάζονται PHYPayloads, δηλαδή τα πλήρη πακέτα σε επίπεδο φυσικού μέσου που μεταδίδονται με διαμόρφωση LoRa. Κάθε PHYPayload περιλαμβάνει τρία βασικά μέρη: το **MHDR** (Message Header) στην αρχή, το **MACPayload** στη συνέχεια και έναν **MIC** (Message Integrity Code) στο τέλος για έλεγχο ακεραιότητας και αυθεντικότητας. Στην Εικόνα 2.16 παρουσιάζεται σχηματικά η γενική μορφή ενός LoRaWAN μηνύματος με τα πεδία του και στη συνέχεια ακολουθεί η ανάλυσή τους [16, 35].



Εικόνα 2.16: Δομή πακέτου LoRaWAN.

[36]

**MHDR (MAC Header):** Είναι η επικεφαλίδα του επιπέδου MAC, μεγέθους 1 byte, που τοποθετείται στην αρχή κάθε PHYPayload. Περιέχει, μεταξύ άλλων, τον τύπο του πλαισίου (**FType**), το οποίο έχει μήκος 3 bit και καθορίζει την κατηγορία/τύπο MAC του πλαισίου. Υπάρχουν 8 τύποι MAC:

- **Join-Request:** Uplink από τη συσκευή για ένταξη μέσω OTAA. Μεταφέρει JoinEUI/AppEUI, DevEUI, DevNonce ώστε ο (Join) Server να παράγει κλειδιά συνεδρίας.
- **Join-Accept:** Downlink απάντηση ένταξης. Περιέχει DevAddr, DLSettings, RxDelay και προαιρετικά CFList. Κρυπτογραφημένο και με MIC.
- **Unconfirmed Data Up:** Uplink δεδομένων χωρίς απαίτηση επιβεβαίωσης (ACK=0).
- **Unconfirmed Data Down:** Downlink δεδομένων χωρίς απαίτηση επιβεβαίωσης.

- **Confirmed Data Up:** Uplink που απαιτεί επιβεβαίωση (ACK από το δίκτυο). Αν δεν ληφθεί ACK στα RX1/RX2 παράθυρα, επιτρέπονται επαναμεταδόσεις.
- **Confirmed Data Down:** Downlink που απαιτεί επιβεβαίωση από τη συσκευή (το ACK επιστρέφεται στο επόμενο uplink μέσω FCtrl.ACK).
- **Rejoin-Request:** (LoRaWAN 1.1) μήνυμα επανένταξης για ανανέωση κλειδιών/επανασυγχρονισμό μετρητών. Υπάρχουν παραλλαγές τύπων (0, 1, 2) για διαφορετικά σενάρια.
- **Proprietary:** Ιδιοταγές πλαίσιο εκτός LoRaWAN MAC (πάνω από LoRa PHY). Χρησιμοποιείται για ειδικές/μη τυπικές εφαρμογές και αγνοείται από τον LoRaWAN Network Server.

Έτσι, από το MHDR μπορεί ο δέκτης να αντιληφθεί εάν, π.χ., το εισερχόμενο πλαίσιο είναι ένα αίτημα σύνδεσης (Join-Request) ή ένα απλό πακέτο δεδομένων. Επιπλέον, περιέχει 2 bit **RFU** (reserved for future use) και ένα πεδίο Major (2 bit) που εντοπίζει τη γενιά πρωτοκόλλου (π.χ. 0 για LoRaWAN v1).

**MACPayload:** Ακολουθεί το MHDR και περιέχει τα κύρια δεδομένα του MAC. Για τα data frames (πλαίσια δεδομένων uplink ή downlink), το MACPayload αποτελείται από τα εξής μέρη: την **FHDR** (Frame Header), ένα πιθανό πεδίο **FPort** και το πραγματικό **FRMPayload** που μπορεί να είναι κρυπτογραφημένο.

**FHDR (Frame Header):** Είναι η κεφαλίδα κάθε πλαισίου δεδομένων, μεγέθους 7 έως 23 byte και περιλαμβάνει τη διεύθυνση συσκευής και τους μετρητές. Συγκεκριμένα περιέχει τη **DevAddr** (32 bit διεύθυνση του node), το **FCtrl** (1 byte με διάφορα control bits), το **FCnt** (Frame Counter, 16 bit) και τυχόν **FOpts** (έως 15 byte). Τα bits του FCtrl έχουν διαφορετική σημασία αναλόγως αν το μήνυμα είναι uplink ή downlink: περιλαμβάνουν το ADR flag, ADR ACK request, bit επιβεβαίωσης ACK κ.ά., καθώς και το μήκος του πεδίου FOpts. Ο μετρητής FCnt αυξάνεται σε κάθε πλαίσιο (ξεχωριστά για uplink και downlink) και χρησιμοποιείται τόσο για anti-replay έλεγχο όσο και στον υπολογισμό του MIC. Το πεδίο FOpts είναι προαιρετικό και μεταφέρει MAC commands ενσωματωμένες στην επικεφαλίδα (π.χ. εντολές LinkADR, DutyCycle κ.ά.) και περιέχει διαδοχικά ζεύγη εντολών μορφής CID, Args, όπου CID είναι 1 byte κωδικός εντολής και Args πιθανά ορίσματα.

**FPort:** Εάν μετά τη FHDR υπάρχει επιπλέον φορτίο δεδομένων, παρεμβάλλεται ένα 1 byte πεδίο FPort που δηλώνει σε ποια θύρα εφαρμογής προορίζεται το FRMPayload. Τιμή  $FPort = 0$  υποδηλώνει ότι το FRMPayload περιέχει μόνο MAC commands (άρα στην ουσία είναι πλαίσιο διαχείρισης και όχι δεδομένων εφαρμογής). Οι τιμές 1-223 μπορούν να χρησιμοποιηθούν σε εφαρμογές από τον χρήστη, η τιμή 224 έχει δεσμευτεί για πειραματικά MAC test frames, ενώ η τιμή 255 είναι RFU (μη χρησιμοποιούμενη).

**FRMPayload:** Είναι το πραγματικό ωφέλιμο φορτίο δεδομένων του πλαισίου, μεταβλητού μήκους (έως 242 byte στο EU868). Μπορεί να είναι κενό ή να περιέχει είτε application payload είτε MAC commands. Εάν  $FPort = 0$ , τότε το FRMPayload περιέχει MAC commands (π.χ. πολλές εντολές μαζί) αντί για δεδομένα εφαρμογής. Διαφορετικά ( $FPort \geq 1$ ), το FRMPayload αντιμετωπίζεται ως δεδομένα προς/από την εφαρμογή και κρυπτογραφείται με το κλειδί εφαρμογής (AppSKey). Στην περίπτωση που έχουμε MAC commands, αυτά

κρυπτογραφούνται με το κλειδί δικτύου (NwkSKey ή στο v1.1 το NwkSEncKey). Όλα τα δεδομένα στο FRMPayload κρυπτογραφούνται για παροχή εμπιστευτικότητας. Η κρυπτογράφηση γίνεται με AES-128 σε λειτουργία CTR (counter mode) χρησιμοποιώντας κατάλληλο κλειδί ανάλογα με την περίπτωση και τον μετρητή πλαισίου FCnt ως μέρος του vector. Η διαδικασία κρυπτογράφησης εγγυάται ότι μόνο ο εξουσιοδοτημένος διακομιστής (Network ή Application αντίστοιχα) μπορεί να αποκωδικοποιήσει το περιεχόμενο.

**MIC (Message Integrity Code):** Στο τέλος κάθε PHYPayload προσαρτάται ένας κώδικας ακεραιότητας MIC μήκους 4 byte (32 bit). Ο MIC υπολογίζεται ως συνάρτηση MAC (AES-CMAC) πάνω σε όλα τα προηγούμενα πεδία του μηνύματος (MHDR + MACPayload), καθώς και μερικά επιπλέον σταθερά bytes που περιλαμβάνουν την κατεύθυνση (uplink/downlink) και τον αντίστοιχο frame counter. Το κλειδί που χρησιμοποιείται για τον MIC διαφέρει ανάλογα με τον τύπο μηνύματος.

Για μηνύματα Join-Request/Join-Accept, ο MIC υπολογίζεται με χρήση του AppKey (ή στο LoRaWAN 1.1 με το NwkKey για το Join-Request).

Για τα μηνύματα Data (uplink ή downlink), ο MIC υπολογίζεται με χρήση του NwkSKey (στο LoRaWAN 1.0) ή με συνδυασμό FNwkSIntKey και SNwkSIntKey (LoRaWAN 1.1) (λεπτομέρειες δίνονται στην επόμενη ενότητα). Σε ουσιαστικό επίπεδο, στο LoRaWAN 1.0 ο ίδιος NwkSKey χρησιμοποιείται τόσο για την επικύρωση των uplink όσο και downlink δεδομένων, ενώ στην έκδοση 1.1 χωρίζεται σε δύο κλειδιά ώστε να μπορούν να συμμετέχουν πολλαπλοί διακομιστές (π.χ. roaming με serving vs home NS) χωρίς να εκτίθεται πλήρως το κλειδί ακεραιότητας. Ο σκοπός του MIC είναι να παρέχει έλεγχο αυθεντικότητας και ακεραιότητας, δηλαδή ο Network Server επαληθεύει ότι μόνο μια συσκευή με γνώση του κατάλληλου κλειδιού θα μπορούσε να έχει δημιουργήσει το εν λόγω πλαίσιο. Αν ο MIC δεν επαληθεύεται, το μήνυμα απορρίπτεται σιωπηλά. Είναι σημαντικό ότι η επαλήθευση MIC γίνεται πριν την αποκρυπτογράφηση των δεδομένων, οπότε το δίκτυο φιλτράρει αποτελεσματικά ψευδή ή αλλοιωμένα πακέτα.

**MAC Commands:** Όπως αναφέρθηκε, εντολές διαχείρισης MAC μπορούν να μεταφέρθούν είτε στο πεδίο FOpts της επικεφαλίδας (μη κρυπτογραφημένες στο LoRaWAN 1.0.x, κρυπτογραφημένες με NwkSEncKey από LoRaWAN 1.1 και μετά) είτε εντός του κρυπτογραφημένου FRMPayload (με FPort = 0). Μερικές σημαντικές εντολές MAC φαίνονται στον Πίνακα 2.5.

Όλες οι MAC εντολές είναι ορισμένες στο πρότυπο με συγκεκριμένους κωδικούς (IDs) και μορφή. Παρέχουν το αναγκαίο εργαλείο στον Network Server για να διαχειρίζεται αποδοτικά το δίκτυο. Για παράδειγμα, μέσω των LinkADR μπορεί να ρυθμίσει μια απομακρυσμένη συσκευή ώστε να χρησιμοποιεί υψηλότερο data rate (μικρότερο SF) αν έχει καλή σύνδεση, μειώνοντας έτσι τον χρόνο στον αέρα και την κατανάλωση (βλ. επόμενη ενότητα για ADR).

Συνολικά, η μορφή του πλαισίου LoRaWAN έχει προβλεφθεί ώστε να μεταφέρει με αποδοτικό τρόπο τόσο τα δεδομένα εφαρμογής όσο και τα απαραίτητα σήματα ελέγχου, μέσα σε ένα πολύ μικρό μήκος payload (συνήθως λίγα byte). Η δομή είναι ευέλικτη, δηλαδή αν δεν υπάρχουν MAC εντολές, το FOpts μπορεί να παραληφθεί ώστε να μεγιστοποιηθεί ο διαθέσιμος χώρος για data. Επίσης, το πρωτόκολλο ορίζει ότι τα πεδία διευθύνσεων και μετρητών είναι διαχειρίσιμα από τον NS, ενώ η εφαρμογή βλέπει μόνο το αποκρυπτογραφημένο περιεχόμενο (δεν ασχολείται με DevAddr ή MIC). Η ενσωματωμένη ασφάλεια στο επίπεδο αυτού

του πλαισίου (κρυπτογράφηση FRMPayload, MIC επί όλων) είναι καθοριστική για να αναπτυχθούν ευρύτατα δίκτυα σε μη προστατευμένες μπάντες συχνοτήτων χωρίς να υπόκεινται σε επιθέσεις υποκλοπής ή τροποποίησης.

MAC Command	Σκοπός	Κατεύθυνση	Μεταφορά
LinkCheck Req/Ans	Έλεγχος συνδεσιμότητας (# πυλών, SNR margin)	Uplink: Req από συσκευή Downlink: Ans από NS	FOpts ί FRMPayload με FPort = 0
LinkADR Req/Ans	Ρύθμιση DR/TXPower/NbTrans, ενεργοποίηση ADR	Downlink: Req από NS Uplink: Ans από συσκευή	FOpts ί FRMPayload με FPort = 0
DutyCycleReq	Θέτει μέγιστο duty cycle για τη συσκευή	Downlink μόνο (από NS)	FOpts ί FRMPayload με FPort = 0
DevStatus Req/Ans	Κατάσταση συσκευής: μπαταρία και SNR margin	Downlink: Req από NS Uplink: Ans από συσκευή	FOpts ί FRMPayload με FPort = 0
NewChannel Req/Ans	Προσθήκη/τροποποίηση καναλιού (Freq, DRRange)	Downlink: Req από NS Uplink: Ans από συσκευή	FOpts ί FRMPayload με FPort = 0
DLChannel Req/Ans	Ορισμός downlink συχνότητας καναλιού	Downlink: Req από NS Uplink: Ans από συσκευή	FOpts ί FRMPayload με FPort = 0
RXParamSetup Req/Ans	Ρυθμίσεις RX1DRoffset, RX2DR, RX2Freq	Downlink: Req από NS Uplink: Ans από συσκευή	FOpts ί FRMPayload με FPort = 0
RXTimingSetupReq	Ρύθμιση καθυστέρησης RX1	Downlink μόνο (από NS)	FOpts ί FRMPayload με FPort = 0
Rekey Ind/Conf	Ανανέωση κλειδιών συνεδρίας (LoRaWAN 1.1)	Downlink: Ind από NS Uplink: Conf από συσκευή	FOpts ί FRMPayload με FPort = 0
Reset Ind/Conf	Δήλωση reset συσκευής (LoRaWAN 1.1)	Uplink: Ind από συσκευή Downlink: Conf από NS	FOpts ί FRMPayload με FPort = 0

Πίνακας 2.5: Συνοπτικός πίνακας βασικών MAC εντολών LoRaWAN.

#### 2.4.4 Ενεργοποίηση συσκευών και Ασφάλεια (OTAA vs ABP)

Η διαδικασία ένταξης μιας συσκευής σε δίκτυο LoRaWAN ονομάζεται ενεργοποίηση (activation). Υπάρχουν δύο μέθοδοι ενεργοποίησης: **Over-The-Air Activation (OTAA)** και **Activation By Personalization (ABP)**. Στην OTAA η συσκευή πραγματοποιεί δυναμική σύνδεση στο δίκτυο μέσω ενός τελετουργικού ανταλλαγής μηνυμάτων (Join Procedure) κατά την οποία λαμβάνει μια προσωρινή διεύθυνση και δημιουργεί κλειδιά συνεδρίας μαζί με το δίκτυο. Η ABP, αντιθέτως, βασίζεται σε στατικά προκαθορισμένες παραμέτρους, δηλαδή η συσκευή είναι προ-προγραμματισμένη με διεύθυνση και κλειδιά και μπορεί να επικοινωνεί αμέσως χωρίς να κάνει join-handshake. Η OTAA θεωρείται ασφαλέστερη και συνιστώμενη

μέθοδος, καθώς παράγει ξεχωριστά κλειδιά για κάθε συνεδρία και επιτρέπει ευκολότερη μεταφορά της συσκευής μεταξύ διαφορετικών δικτύων, ενώ η ABP ενέχει κινδύνους (στατικά κλειδιά που μπορεί να αποκαλυφθούν) και έλλειψη ευελιξίας (η συσκευή «κλειδώνεται» σε συγκεκριμένο δίκτυο και η αλλαγή παρόχου απαιτεί χειροκίνητη αναδιαμόρφωση) [16, 37].

#### 2.4.4.1 Over-The-Air Activation (OTAA)

Πριν ξεκινήσει η OTAA, σε κάθε συσκευή είναι φορτωμένα ορισμένα αναγνωριστικά και κλειδιά: συγκεκριμένα ένα **DevEUI** (παγκόσμια μοναδικό 64-bit αναγνωριστικό συσκευής), ένα **AppEUI/JoinEUI** (64-bit αναγνωριστικό της εφαρμογής ή του Join Server που θα την χειρίστει) και ένα μυστικό κλειδί **AppKey** (128-bit AES κλειδί, γνωστό μόνο στη συσκευή και στον αντίστοιχο διακομιστή, είτε Network Server παλαιότερα, είτε Join Server στα νεότερα δίκτυα).

Η διαδικασία OTAA στα LoRaWAN 1.0.x και 1.1.x περιλαμβάνει δύο μηνύματα MAC:

- **Join-Request (uplink):** Η συσκευή στέλνει ένα αίτημα σύνδεσης. Το μήνυμα αυτό περιέχει τα πεδία **AppEUI**, **DevEUI** και ένα **DevNonce**. Το DevNonce είναι μια τυχαία δυαδική τιμή 2 byte που η συσκευή επιλέγει κάθε φορά που κάνει join. Ο Network/Join Server αποθηκεύει το τελευταίο DevNonce που έχει δει από τη συγκεκριμένη συσκευή ώστε να αποτρέψει επιθέσεις επανάληψης (αν λάβει ξανά Join-Request με ίδιο DevNonce, το απορρίπτει). Το Join-Request δεν είναι κρυπτογραφημένο (μεταδίδεται σε ένα από τα ειδικά κανάλια join της εκάστοτε περιοχής, π.χ. 868.10, 868.30, 868.50MHz στην Ευρώπη), αλλά προστατεύεται με MIC που υπολογίζεται με το AppKey [16, 37].

8 bytes	8 bytes	2 bytes
AppEUI	DevEUI	DevNonce

Πίνακας 2.6: Πεδία Join-Request στο LoRaWAN 1.0.

8 bytes	8 bytes	2 bytes
JoinEUI	DevEUI	DevNonce

Πίνακας 2.7: Πεδία Join-Request στο LoRaWAN 1.1.

- **Join-Accept (downlink):** Αν το δίκτυο αποδεχτεί την αίτηση, αποστέλλει ένα μήνυμα Join-Accept. Στις εκδόσεις 1.0.x αυτό το μήνυμα παράγεται από τον Network Server (ή Application Server) ενώ στις 1.1+ παράγεται από τον Join Server. Το Join-Accept περιλαμβάνει κρίσιμες πληροφορίες: ένα **AppNonce** (24-bit τυχαίο που παράγει το δίκτυο για τη συσκευή), το **NetID** του δικτύου, τη νέα **DevAddr** που εκχωρείται στη συσκευή, το **DLSettings** (ρυθμίσεις για τα παράθυρα λήψης, π.χ. αρχικό κανάλι RX2), το **RxDelay** (καθυστέρηση RX1) και πιθανώς ένα **CFLList** (λίστα επιπλέον καναλιών που θα χρησιμοποιεί η συσκευή).

3 bytes	3 bytes	4 bytes	1 byte	1 byte	16 bytes (optional)
AppNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

Πίνακας 2.8: Πεδία Join-Accept στο LoRaWAN 1.0.

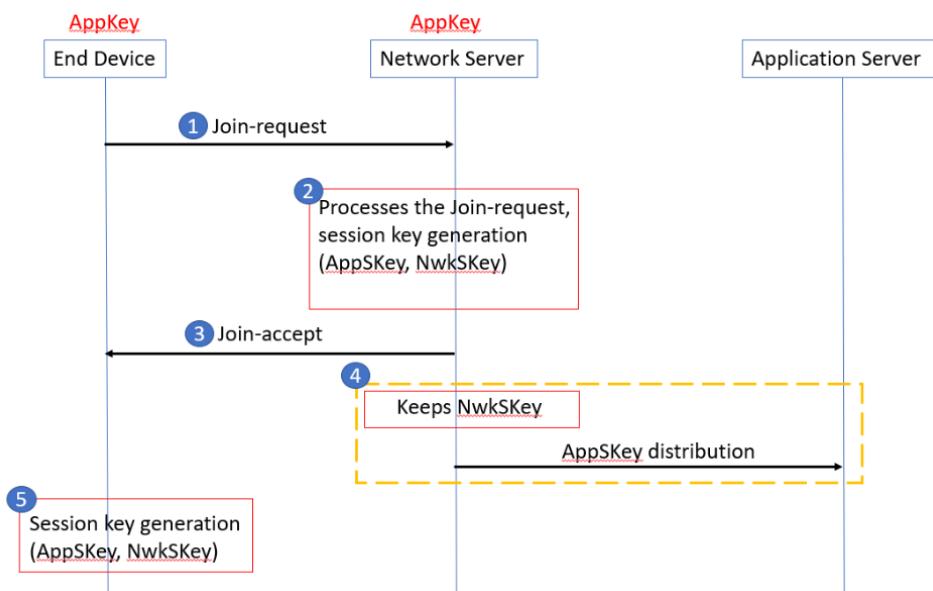
3 bytes	3 bytes	4 bytes	1 byte	1 byte	16 bytes (optional)
JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

Πίνακας 2.9: Πεδία Join-Accept στο LoRaWAN 1.1.

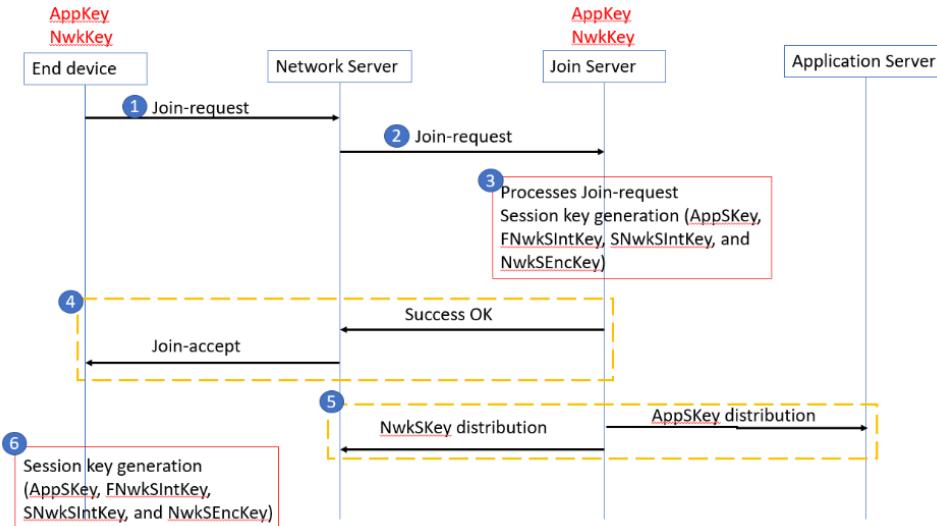
Το Join-Accept κρυπτογραφείται πριν σταλεί. Συγκεκριμένα, στην έκδοση 1.0 με το AppKey, ενώ στην 1.1 με ξεχωριστό κλειδί ανάλογα την περίπτωση (NwkKey αν προκλήθηκε από Join-Request, ή JSEncKey αν ήταν από Rejoin). Φέρει επίσης MIC (4-byte) που υπολογίζεται με χρήση του AppKey ή του JSIntKey αντίστοιχα. Όταν η συσκευή λάβει το Join-Accept, το αποκρυπτογραφεί (με χρήση του δικού της AppKey) και ελέγχει τον MIC και την τιμή του AppNonce (ότι είναι νέα, για αποτροπή παλιού accept). Εφόσον όλα είναι έγκυρα, προχωρά στον υπολογισμό των κλειδιών συνεδρίας. Στο LoRaWAN 1.0 χρησιμοποιείται το AppKey για να παραχθούν δύο 128-bit κλειδιά, το NwkSKey (Network Session Key) και το AppSKey (Application Session Key). Οι ακριβείς τύποι δίνονται στο πρότυπο:

$$NwkSKey = \text{AES}_{128}(\text{AppKey}, 0x01 \| \text{AppNonce} \| \text{NetID} \| \text{DevNonce}) ,$$

$$\text{AppSKey} = \text{AES}_{128}(\text{AppKey}, 0x02 \| \text{AppNonce} \| \text{NetID} \| \text{DevNonce}) .$$



Εικόνα 2.17: Ροή μηνυμάτων για ενεργοποίηση OTAA στο LoRaWAN 1.0.



Εικόνα 2.18: Ροή μηνυμάτων για ενεργοποίηση OTAA στο LoRaWAN 1.1.

[16]

Η συσκευή και ο Network Server κατ' αυτόν τον τρόπο μοιράζονται το ίδιο NwkSKey και η συσκευή με τον Application Server το ίδιο AppSKey (στην πράξη ο NS κρατά το NwkSKey και μεταβιβάζει το AppSKey στον Application Server). Από τη στιγμή αυτή, όλες οι επικοινωνίες θα χρησιμοποιούν αυτά τα κλειδιά. Ο NS θα χρησιμοποιεί το NwkSKey για να ελέγχει/υπογράφει τα επόμενα μηνύματα και ο AS το AppSKey για να αποκρυπτογραφεί τα δεδομένα. Η συσκευή επίσης μηδενίζει τους frame counters  $FCntUp = 0$ ,  $FCntDown = 0$  και ξεκινά κανονική ανταλλαγή data frames. Στο LoRaWAN 1.1 η διαδικασία έχει ορισμένες τροποποιήσεις που ενισχύουν την ασφάλεια:

- Χρησιμοποιούνται δύο διακριτά Root Keys: το AppKey (για την εφαρμογή) και ένα NwkKey (για το δίκτυο). Ο Join Server (ή το δίκτυο) έχει και τα δύο, αλλά τα διαμοιράζει ξεχωριστά.
- To Join-Request προστατεύεται με MIC υπολογισμένο με το NwkKey αντί AppKey, ώστε μόνο ο Join Server (που έχει το NwkKey) να το επαληθεύσει.
- To Join-Accept κρυπτογραφείται με NwkKey (αν ανταποκρίνεται σε Join-Request) ή JSEncKey (αν ανταποκρίνεται σε Rejoin).
- Παράγονται τέσσερα κλειδιά συνεδρίας: το AppSKey (για εφαρμογή) που πλέον προκύπτει μόνο από το AppKey και τρία κλειδιά δικτύου από το NwkKey: FNwkSIntKey (Forwarding NS Integrity), SNwkSIntKey (Serving NS Integrity) και NwkSEncKey (Encryption for MAC). Σε ένα μη roaming σενάριο (συσκευή και NS στο ίδιο δίκτυο) τα FNwkSInt και SNwkSInt συμπίπτουν, αλλά η διάκριση επιτρέπει roaming μεταξύ ενός Home NS και Serving NS χωρίς ανταλλαγή κλειδιών ακεραιότητας.
- Ο μηχανισμός πλαισίων data στο 1.1 αλλάζει ως προς τον MIC. Αντί ενός MIC με ένα κλειδί, χρησιμοποιούνται δύο συνιστώσες (cmacF και cmacS) υπολογισμένες με FNwkSIntKey και SNwkSIntKey αντίστοιχα και συνδυάζονται (2 byte από κάθε μία)

για να σχηματίσουν το 4-byte MIC. Αυτό εξασφαλίζει ότι ένας «πρωθητικός» NS σε roaming (fNS) που έχει το FNwkSIntKey μπορεί να ελέγξει τη μισή υπογραφή χωρίς να μπορεί να την παράγει πλήρως (δεν έχει το SNwkSIntKey που το έχει μόνο ο server του home δικτύου).

- Επίσης, στην έκδοση 1.1, αν μια συσκευή κάνει ABP έχει την απαίτηση να μην μηδενίζει τους frame counters μετά από reset (σε αντίθεση με 1.0 όπου το έκανε), ώστε να αποφεύγονται replay attacks λόγω επανεκκίνησης. Σε περίπτωση που γίνει reset, εισάγεται το MAC command ResetInd/ResetConf ώστε η συσκευή να ενημερώσει τον NS και να επανασυγχρονιστούν χωρίς κίνδυνο ασφάλειας.

#### 2.4.4.2 Activation By Personalization (ABP)

Στην περίπτωση της ενεργοποίησης με προσωποποίηση, δεν υπάρχει ανταλλαγή μηνυμάτων join. Αντίθετα, η συσκευή έρχεται προρυθμισμένη με όλα τα απαραίτητα στοιχεία. Συγκεκριμένα, είναι αποθηκευμένα στη μνήμη της μια DevAddr καθώς και τα κλειδιά NwkSKey και AppSKey (στο LoRaWAN 1.0) ή τα 4 session keys (AppSKey, FNwkSIntKey, SNwkSIntKey, NwkSEncKey στο LoRaWAN 1.1). Τα ίδια κλειδιά και η DevAddr είναι καταχωρημένα χειροκίνητα και στον Network Server (και Application Server) εκ των προτέρων. Έτσι, η συσκευή μόλις ανοίξει μπορεί άμεσα να σέλνει data frames χρησιμοποιώντας αυτά τα στοιχεία, χωρίς διαδικασία OTAA. Προφανώς, αυτό σημαίνει ότι όλα τα ABP nodes πρέπει να έχουν μοναδικά κλειδιά, μιας και το πρότυπο απαιτεί κάθε συσκευή να έχει δικό της ζεύγος NwkSKey/AppSKey για λόγους ασφαλείας (ώστε αν παραβιαστεί μία, να μην επιτρεπατούν οι άλλες). Η ABP είναι χρήσιμη σε περιπτώσεις που το join μπορεί να είναι δύσκολο ή σε δοκιμές/εργαστήρια όπου θέλουμε να παρακάμψουμε το overhead [16] [37]. Ωστόσο, εμφανίζει αρκετά μειονεκτήματα:

- **Ασφάλεια:** Τα κλειδιά είναι στατικά και συνήθως ενσωματωμένα στο firmware. Συνεπώς, αν κάποιος αποκτήσει φυσική πρόσθαση στη συσκευή, μπορεί να τα εξάγει. Επίσης, δεν αλλάζουν ποτέ, άρα μια διαρροή κλειδιού εκθέτει όλη τη μετέπειτα επικοινωνία.
- **Διαχείριση:** Δεν υπάρχει εύκολος τρόπος ανανέωσης κλειδιών ή DevAddr. Αν χρειαστεί να αλλάξει δίκτυο η συσκευή (άλλον πάροχο ή δίκτυο), πρέπει να αναπρογραμματιστεί με νέα στοιχεία (ενώ με OTAA απλώς συνδέεται στο νέο δίκτυο).
- **Frame Counters:** Στο LoRaWAN 1.0, μια ABP συσκευή αν κάνει επανεκκίνηση μηδενίζει το FCnt, αλλά ο NS κρατά την προηγούμενη τιμή, γεγονός που μπορεί να δημιουργήσει συγχύσεις (αν ο NS δει ξανά μικρότερο counter θα απορρίπτει τα μηνύματα ως replay). Συνήθως λύνεται με ρυθμίσεις στον NS (disable FCnt check), κάτι που όμως μειώνει την ασφάλεια. Στο LoRaWAN 1.1, όπως προαναφέρθηκε, απαιτείται να μη μηδενίζει η συσκευή τους counters ή να σέλνει ResetInd MAC command.
- **Roaming:** Μια ABP συσκευή είναι δεσμευμένη σε ένα NetID δικτύου. Το LoRaWAN 1.1 εισάγει μεν την έννοια του roaming και για ABP (Passive Roaming), αλλά χρειάζεται η συνεργασία δικτύων και δεν είναι τόσο απλό όσο μια OTAA Rejoin.

Λόγω των παραπάνω, η OTAA προτιμάται σαφώς στην πράξη. Η ABP χρησιμοποιείται μόνο σε ειδικές περιπτώσεις ή για απλούστευση κατά το στάδιο ανάπτυξης/πρωτοτυποποίησης.



Εικόνα 2.19: Προ-διαμοιρασμός του *DevAddr* και των *session keys* για ενεργοποίηση ABP στο LoRaWAN 1.0.

[16]



Εικόνα 2.20: Προ-διαμοιρασμός του *DevAddr* και των *session keys* για ενεργοποίηση ABP στο LoRaWAN 1.1.

[16]

#### 2.4.4.3 Ασφάλεια E2E και κρυπτογραφία

Το LoRaWAN στηρίζεται σε συμμετρική κρυπτογραφία AES-128 για να πετύχει δύο βασικούς στόχους: α) εμπιστευτικότητα δεδομένων και β) ακεραιότητα/αυθεντικότητα μηνυμάτων. Όπως εξηγήθηκε, η εμπιστευτικότητα επιτυγχάνεται κρυπτογραφώντας το πεδίο FRMPayload κάθε data frame με το AppSKey (για εφαρμογές) ή το NwkSEncKey (για MAC commands) σε CTR mode. Αυτό εξασφαλίζει ότι ακόμα και αν κάποιος υποκλέψει το ασύρματο σήμα, δεν μπορεί να διαβάσει τα δεδομένα χωρίς το κλειδί. Η ακεραιότητα/αυ-

Θεντικότητα επιτυγχάνεται με τον έλεγχο MIC που υπογράφει κάθε μήνυμα με κλειδί που γνωρίζει μόνο το δίκτυο (NwkSKey / FNwkSIntKey κ.λπ.), αποτρέποντας τόσο την τροποποίηση πακέτων όσο και την εισαγωγή ψεύτικων από τρίτους [33].

Σημαντικό είναι ότι τα κλειδιά αυτά είναι ξεχωριστά ανά συσκευή και ανά συνεδρία. Επίσης, δεν μεταδίδονται ποτέ «καθαρά» πάνω από τον αέρα, αλλά παράγονται τοπικά από τις δύο μεριές (συσκευή/JS και NS/AS) και αποθηκεύονται με ασφαλή τρόπο. Αυτή η φιλοσοφία end-to-end encryption φαίνεται και στην 2.10, όπου το φορτίο παραμένει κρυπτογραφημένο από τη στιγμή που φεύγει από τη συσκευή μέχρι να φτάσει στον Application Server. Ο Network Server δεν γνωρίζει (ούτε χρειάζεται να γνωρίζει) το περιεχόμενο των εφαρμοστικών δεδομένων, καθώς λειτουργεί απλώς ως αγωγός. Αυτό είναι ιδιαίτερα κρίσιμο σε εφαρμογές όπως π.χ. έξυπνοι μετρητές ρεύματος, όπου τα δεδομένα κατανάλωσης πρέπει να προστατεύονται ιδιωτικά μέχρι τον πάροχο ενέργειας που τα διαχειρίζεται (το LoRaWAN το διασφαλίζει ήδη από το στάδιο του σχεδιασμού).

Τέλος, αξίζει να αναφερθεί ότι πέρα από τα εσωτερικά πρότυπα του LoRaWAN, υπάρχει και συνεισφορά από την IETF για διασύνδεση των δικτύων LoRaWAN με το διαδίκτυο σε επίπεδο IP. Συγκεκριμένα, έχει οριστεί ένα πρότυπο συμπίεσης επικεφαλίδων IPv6/UDP γνωστό ως **SCHC (Static Context Header Compression)** για χρήση πάνω από LoRaWAN, επιτρέποντας αποδοτική μεταφορά πακέτων IPv6 μέσω LoRaWAN με ελάχιστη επιβάρυνση. Το πρότυπο αυτό δημοσιεύτηκε ως RFC 9011 (2021) και ουσιαστικά ορίζει πώς μπορούν να διαμοιράζονται κανόνες συμπίεσης μεταξύ τερματικού και δικτύου ώστε ακόμη και τα μικρά payloads του LoRaWAN να μεταφέρουν δεδομένα IPv6 όταν απαιτείται (π.χ. σε βιομηχανικές εφαρμογές IPv6 sensor networking). Αυτό υπογραμμίζει τη modular αρχιτεκτονική, δηλαδή το LoRaWAN μπορεί να θεωρηθεί ως layer 2.5 που κουβαλά αν θέλουμε και υψηλότερα πρωτόκολλα με ειδική προσαρμογή [38].

#### 2.4.5 Ρυθμός μετάδοσης, ADR, χρόνος στον αέρα (ToA) και περιορισμοί εκπομπής

Όπως έχει ήδη αναφερθεί σε προγενέστερες ενότητες, το φυσικό layer LoRa (CSS) που χρησιμοποιεί το LoRaWAN επιτυγχάνει διάφορους ρυθμούς δεδομένων (bit rates) μεταβάλλοντας παραμέτρους όπως ο Spreading Factor (SF) και το εύρος ζώνης. Στο LoRaWAN αυτοί οι ρυθμοί έχουν κβαντιστεί σε διακριτά Data Rates (DR), τυπικά αριθμημένα (π.χ. DR0 - DR5 για την EU) και αντιστοιχούν σε συγκεκριμένες ρυθμίσεις (π.χ. DR0 = SF12/125kHz - 250bps, ..., DR5 = SF7/125kHz - 5469bps στο EU863-870). Ο χρόνος στον αέρα (Time-on-Air, ToA) ενός πλαισίου LoRa αυξάνεται δραματικά όσο μειώνεται ο ρυθμός (υψηλότερο SF), ένα μήνυμα 50 byte στο SF7 μπορεί να διαρκέσει 50 ms, ενώ στο SF12 μπορεί να διαρκέσει 1.6 δευτερόλεπτα. Η επιλογή του data rate συνεπώς επηρεάζει τόσο την κατανάλωση ενέργειας της συσκευής (μεγάλος ToA = πολλή ενέργεια για εκπομπή) όσο και τη συνολική χωρητικότητα του δικτύου (καθώς το κανάλι δεσμεύεται για περισσότερη ώρα, εμποδίζοντας άλλες μεταδόσεις). Το LoRaWAN αντιμετωπίζει αυτό το ζήτημα με δύο τρόπους: α) μέσω του μηχανισμού Adaptive Data Rate (ADR) και β) μέσω κανονιστικών περιορισμών εκπομπής (duty cycle κ.λπ.) που επιβάλλονται στις μη αδειοδοτημένες συχνότητες [16].

#### 2.4.5.1 Adaptive Data Rate (ADR)

Το ADR είναι ένας μηχανισμός του πρωτοκόλλου που επιτρέπει στο δίκτυο να βελτιστοποιεί δυναμικά τον ρυθμό δεδομένων και την ισχύ μετάδοσης μιας συσκευής, βασιζόμενο στις συνθήκες ζεύξης. Όταν μια συσκευή έχει το ADR ενεργοποιημένο (bit ADR=1 στα uplinks της) ουσιαστικά δηλώνει στον Network Server ότι είναι στατική ή έχει σταθερές συνθήκες και επιτρέπει στο δίκτυο να ρυθμίσει τις παραμέτρους της. Ο NS τότε συλλέγει μετρήσεις από τα πρόσφατα uplinks (μέχρι 20 τελευταία) όπως το SNR σε πολλαπλές πύλες, τον αριθμό gateways που την «άκουσαν» κ.λπ. Με βάση αυτές, υπολογίζει ένα περιθώριο (margin) σήματος για τη συσκευή. Για παράδειγμα, αν λαμβάνεται με SNR πολύ πάνω από το ελάχιστο, αυτό σημαίνει ότι η συσκευή μπορεί να ανεβάσει το data rate (μικρότερο SF) ή/και να χαμηλώσει την ισχύ της χωρίς να χαθεί η επικοινωνία. Κατόπιν, ο NS στέλνει μία ή περισσότερες εντολές LinkADRReq προς τη συσκευή, ορίζοντας νέο SF, bandwidth, ισχύ και ενδεχομένως μάσκα καναλιών. Η συσκευή εκτελεί τις ρυθμίσεις και απαντά με LinkADRAAns (που επιβεβαιώνει ή απορρίπτει). Μέσω αυτού του feedback loop, το δίκτυο τείνει να ρυθμίσει όλες τις συσκευές στο ταχύτερο δυνατό data rate που επιτρέπει η απόστασή τους. Ειδικότερα, όσες συσκευές είναι κοντά σε πύλη θα μειώσουν σε SF7, εξοικονομώντας χρόνο στον αέρα και μπαταρία, ενώ όσες είναι μακριά θα παραμείνουν σε υψηλότερο SF για μεγαλύτερη αξιοπιστία. Σε περίπτωση που μια συσκευή μετακινηθεί ή αλλάξουν οι συνθήκες (π.χ. εμποδισμός του σήματος), η ίδια μπορεί να αντιληφθεί υποθάμηση (δεν λαμβάνει καθόλου downlinks για αρκετή ώρα) και να απενεργοποιήσει προσωρινά το ADR (ADR flag=0) οπότε θα υποβιβαστεί αυτόματα στο πιο «ασφαλές» χαμηλό data rate έως ότου οι συνθήκες σταθεροποιηθούν ξανά.

Ο αλγόριθμος ADR δεν καθορίζεται πλήρως στο πρότυπο παρά μόνο ως σύσταση. Για παράδειγμα, η Semtech έχει δημοσιεύσει έναν απλό αλγόριθμο που λαμβάνει τον καλύτερο SNR από τα τελευταία 20 uplinks και αυξάνει βαθμιαία το data rate μέχρι το SNR margin να πέσει κάτω από κάποιο threshold. Συστήματα όπως το The Things Stack ακολουθούν τέτοιες μεθοδολογίες, εφαρμόζοντας μικρές υστερήσεις για να αποφεύγονται οι ταλαντώσεις (π.χ. να μη στέλνουν συνεχώς ADRReq αν η συσκευή αρνείται ή αν τα στοιχεία είναι αμφίβολα).

Εν γένει, το ADR είναι εξαιρετικά χρήσιμο για στατικές συσκευές (π.χ. αισθητήρες σε πάγια θέση) μιας και βελτιστοποιεί αυτόματα την κατανάλωσή τους και βελτιώνει τη χωρητικότητα του δικτύου. Για κινητές συσκευές (π.χ. ιχνηλάτες σε οχήματα) το ADR μπορεί να προκαλέσει αστάθεια. Συνιστάται να απενεργοποιείται ή να χρησιμοποιείται μόνο όταν η συσκευή αντιληφθεί ότι παραμένει στάσιμη για αρκετό χρόνο. Σε κάθε περίπτωση, η τελική απόφαση για χρήση ADR ή όχι λαμβάνεται από τη συσκευή (το application μπορεί να το ενεργοποιήσει/απενεργοποιήσει αναθέτοντας την κατάλληλη τιμή στο ADR bit), ωστόσο στα περισσότερα δίκτυα IoT προτιμάται να είναι ενεργό για να επωφελούνται οι κόμβοι [16].

Στην περιοχή EU863-870 (EU868), το μέγιστο ωφέλιμο App Payload που μπορεί να στείλει μια συσκευή εξαρτάται από τον συνδυασμό Data Rate (DR), Spreading Factor (SF) και Bandwidth (BW). Το πρότυπο ορίζει κατηγορίες DR0,..., DR7 με αντίστοιχα SF/BW, και για την «repeater-compatible» λειτουργία (που χρησιμοποιεί το The Things Stack, βλ. Ενότητα 3.1) τα ανώτατα μεγέθη ωφέλιμου είναι:

Data Rate	Modulation / Rate	Max MACPayload	Max App Payload
DR0	LoRa SF12 / 125 kHz	59	51
DR1	LoRa SF11 / 125 kHz	59	51
DR2	LoRa SF10 / 125 kHz	59	51
DR3	LoRa SF9 / 125 kHz	123	115
DR4	LoRa SF8 / 125 kHz	230	222
DR5	LoRa SF7 / 125 kHz	230	222
DR6	LoRa SF7 / 250 kHz	230	222
DR7	FSK 50 kbps	230	222

Πίνακας 2.10: EU863-870 (EU868): μέγιστα μεγέθη MACPayload και ωφέλιμου App Payload, repeater-compatible. To App Payload υπολογίζεται από MACPayload με αφαίρεση του LoRaWAN overhead (θεωρώντας άδειο FOpts).

[12]

#### 2.4.5.2 Περιορισμοί Duty Cycle και κανονισμοί περιοχής

Δεδομένου ότι το LoRaWAN λειτουργεί σε ελεύθερες (μη αδειοδοτημένες) ζώνες ISM, η εκπομπή των συσκευών διέπεται από κανονισμούς που αποσκοπούν στην αποφυγή κατάχρησης του φάσματος. Στην Ευρώπη, ισχύει το πρότυπο ETSI EN 300.220, το οποίο για την μπάντα 868MHz επιβάλλει μέγιστο Duty Cycle 1% στις περισσότερες υποζώνες, με ορισμένες εξαιρέσεις (π.χ. 0.1% σε 869.40-869.65MHz όπου επιτρέπεται 10%). Το Duty Cycle ορίζεται ως ο λόγος χρόνου εκπομπής προς το συνολικό χρόνο σε ένα κανάλι. 1% duty cycle οντιμαίνει ότι μια συσκευή μπορεί να εκπέμπει το πολύ 36 δευτερόλεπτα ανά ώρα σε μια συγκεκριμένη συχνότητα. Αν εκπέμψει συνεχόμενα π.χ. για 3 δευτερόλεπτα, θα πρέπει να περιμένει 297 δευτερόλεπτα πριν επανεκπέμψει στο ίδιο κανάλι (για να διατηρήσει τον λόγο 1/100). Ο περιορισμός αυτός, συνδυαζόμενος με τους μεγάλους χρόνους στον αέρα στα χαμηλά data rates, ουσιαστικά περιορίζει τον ρυθμό πακέτων που μπορεί να στέλνει ένας κόμβος. Στα δίκτυα LoRaWAN είναι σύνηθες να στέλνουν οι αισθητήρες δεδομένα ανά λίγα λεπτά (π.χ. ανά 5 ή 15 λεπτά) ώστε να τηρούν άνετα το duty cycle. Ο Network Server μπορεί να βοηθάει στον έλεγχο αυτό, για παράδειγμα, μέσω ADR, κρατώντας τον ToA χαμηλό ή μέσω του MAC command DutyCycleReq, επιβάλλοντας μικρότερο duty cycle από το νομικό όριο σε συγκεκριμένες περιπτώσεις [37, 16].

Στις ΗΠΑ και σε άλλες περιοχές (902-928MHz band), αντί για duty cycle, ισχύουν κανονισμοί dwell time (μέγιστη διάρκεια συνεχούς εκπομπής περίπου 400ms) και Frequency Hopping (υποχρέωση αλλαγής συχνότητας σε κάθε πακέτο). Το LoRaWAN σε αυτές τις περιοχές χρησιμοποιεί μεγάλο αριθμό καναλιών (π.χ. 64 uplink channels στο US915) και μια τυχαία κατανομή των πακέτων σε αυτά. Έτσι διασφαλίζεται ότι πληροί τους κανόνες και κατανέμει τη χρήση του φάσματος. Σε κάποιες χώρες (π.χ. στην Ιαπωνία) επιβάλλεται μηχανισμός LBT (Listen-Before-Talk) αντί του duty cycle, όπου η συσκευή οφείλει να ακούσει ότι το κανάλι είναι καθαρό πριν εκπέμψει. Το LoRaWAN Regional Parameters προσαρμόζει αυτές τις λεπτομέρειες ανά χώρα.

Το έγγραφο **LoRaWAN Regional Parameters** από τη LoRa Alliance ορίζει για κάθε περιοχή τα διαθέσιμα σχέδια συχνοτήτων (διαύλους) και τις ειδικές ρυθμίσεις. Για παράδειγμα,

το EU868 σχέδιο ορίζει 3 βασικά κανάλια (867.1, 867.3, 867.5MHz) που πάντα πρέπει να υποστηρίζει μια συσκευή και duty cycle 1% σε όλη την μπάντα 863-870 εκτός ορισμένων μικρών υποζωνών. Το US915 σχέδιο ορίζει 64 κανάλια uplink με hop κάθε φορά και υποδιαιρεση σε 8 sub-band όπου κάθε sub-band έχει όριο 0.4s dwell per channel. Το AS923 (Ασία) ορίζει μια κοινή ομάδα 16 καναλιών για πολλές χώρες στην περιοχή, αλλά σε μερικές (π.χ. Νέα Ζηλανδία και Ιαπωνία) διαφοροποιείται ως προς τα LBT και duty cycle [12].

Εν κατακλείδει, το LoRaWAN προσπαθεί να προσφέρει έναν ενοποιημένο τρόπο επικοινωνίας. Συγκεκριμένα, η συσκευή σε κάθε περιοχή έχει μια λίστα προκαθορισμένων καναλιών και γνωρίζει τις ανώτατες επιτρεπόμενες ισχείς εκπομπής (π.χ. 14dBm στην Ευρώπη, 30dBm στις ΗΠΑ) και τη μέγιστη διάρκεια πακέτου. Η συμμόρφωση με αυτούς τους περιορισμούς είναι υποχρεωτική, οπότε κάθε node πρέπει να προγραμματίζεται έτσι ώστε να μην υπερβαίνει τα όρια (π.χ. να μην στέλνει πολύ συχνά ώστε να μην υπερβεί το duty cycle). Πολλές LoRaWAN συσκευές διαθέτουν εσωτερικό έλεγχο duty cycle, όπως για παράδειγμα το module RN2483 της Microchip δεν θα επιτρέψει νέα εκπομπή αν το κανάλι δεν είναι ελεύθερο ως προς το 1% (θα επιστρέψει σφάλμα no\_free\_ch). Αυτό προστατεύει το σύστημα από παραβίαση των κανονισμών ακόμα και αν η εφαρμογή προγραμματίστηκε λάθος.



## Κεφάλαιο 3

# Τεχνολογική στοίβα λογισμικού του συστήματος

---

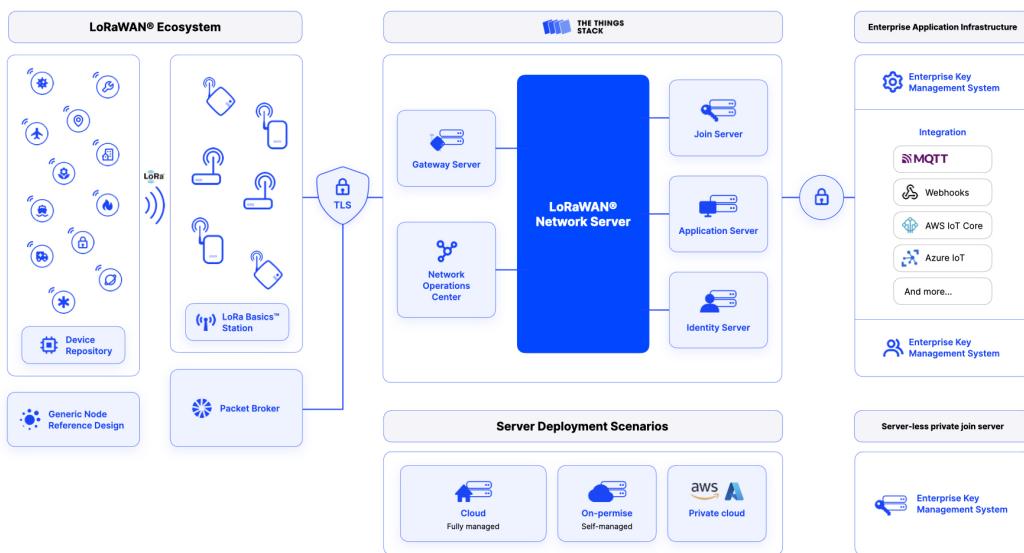
Σε αυτό το κεφάλαιο παρουσιάζονται οι κύριες τεχνολογίες και τα εργαλεία που αποτέλεσαν βασικά μέρη του συστήματος που υλοποιήθηκε. Συγκεκριμένα, οι τεχνολογίες αυτές είναι:

- **The Things Stack:** μία στοίβα (stack) LoRaWAN δικτύου ανοικτού κώδικα, το οποίο περιλαμβάνει τον LoRaWAN Network Server που είναι απαραίτητος για τη δημιουργία του δικτύου LoRaWAN, καθώς και άλλους διακομιστές όπως τον Application Server, τον Gateway Server κ.ά.
- **LoRa Basics Station:** το σύγχρονο λογισμικό packet forwarder που χρησιμοποιήθηκε στο LoRa gateway για ασφαλή σύνδεση με τον Network Server.
- **Docker:** η ανοιχτού κώδικα πλατφόρμα για διαχείρηση software containers με την οποία γίνεται η φιλοξενία και εκτέλεση όλων των υπηρεσιών του συστήματος, δηλαδή του The Things Stack, του LoRa Basics Station, καθώς και της εφαρμογής (backend και frontend) σε απομονωμένα περιβάλλοντα.
- **Spring Boot:** το framework της Java που χρησιμοποιήθηκε για την ανάπτυξη της διαδικτυακής εφαρμογής στον διακομιστή (backend) του συστήματος.
- **React JS:** η βιβλιοθήκη JavaScript που χρησιμοποιήθηκε για την υλοποίηση του διαδικτυακού περιβάλλοντος χρήστη (frontend) της εφαρμογής.
- **PostgreSQL:** το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων του συστήματος.

### 3.1 The Things Stack

To The Things Stack (TTS) είναι μια ανοικτού κώδικα στοίβα λογισμικού για δίκτυα LoRaWAN, κατάλληλη για την υποστήριξη τόσο μεγάλων, παγκόσμιων και γεωγραφικά κατανεμημένων δικτύων όσο και μικρότερων ιδιωτικών εγκαταστάσεων. Η αρχιτεκτονική του ακολουθεί το πρότυπο αναφοράς του LoRaWAN και διασφαλίζει τη διαλειτουργικότητα και τη συμμόρφωση με τις προδιαγραφές του πρωτοκόλλου. Ουσιαστικά, το The Things Stack

συνιστά τον «πυρήνα» ενός δικτύου LoRaWAN, καθώς είναι υπεύθυνο για τη διασύνδεση, τη διαχείριση και την παρακολούθηση όλων των συσκευών, των gateways και των εφαρμογών των τελικών χρηστών του δικτύου. Κύριος στόχος του είναι να εξασφαλίσει την ασφαλή, επεκτάσιμη και αξιόπιστη δρομολόγηση των δεδομένων από τους αισθητήρες προς τις εφαρμογές και αντίστροφα, εφαρμόζοντας τους μηχανισμούς ασφαλείας και ελέγχου πρόσθασης του LoRaWAN.



Εικόνα 3.1: Αρχιτεκτονική του The Things Stack με τα βασικά του δομικά στοιχεία.  
[39]

To The Things Stack ακολουθεί μια αρχιτεκτονική μικροϋπηρεσιών (microservice architecture) όπου επιμέρους υπηρεσίες συνεργάζονται μέσω σαφώς ορισμένων διεπαφών. Οι κύριες συνιστώσες του φαίνονται σχηματικά στην Εικόνα 3.1 και συνοψίζονται ως εξής:

**Gateway Server (GS):** Διαχειρίζεται τις συνδέσεις με τα LoRaWAN gateways, υποστηρίζοντας πολλαπλά πρωτόκολλα διασύνδεσης όπως το κλασικό UDP packet forwarder, το νεότερο LoRa Basics Station, καθώς και τα MQTT ή gRPC. Ο Gateway Server λαμβάνει τα uplink πακέτα από κάθε gateway και τα προωθεί στον Network Server, ενώ προγραμματίζει και αποστέλλει τα downlink πακέτα προς τα κατάλληλα gateway. Επιπλέον, φροντίζει για την ασφαλή αυθεντικοποίηση των gateways και την απομακρυσμένη διαχείρισή τους [39].

**Network Server (NS):** Υλοποιεί τον «πυρήνα» του πρωτοκόλλου LoRaWAN σε επίπεδο δικτύου. Αυτό περιλαμβάνει την επεξεργασία MAC εντολών, την εφαρμογή των περιφερειακών παραμέτρων (π.χ. περιορισμοί συχνοτήτων) και τον μηχανισμό Προσαρμοστικού Ρυθμού Δεδομένων (ADR) για τη βελτιστοποίηση της μετάδοσης. Ο Network Server επαληθεύει την αυθεντικότητα και ακεραιότητα των μηνυμάτων από τις συσκευές, απορρίπτει διπλότυπα πακέτα (σε περίπτωση που το ίδιο uplink ληφθεί από πολλαπλά gateway) και επιλέγει το βέλτιστο gateway για αποστολή κάθε downlink. Επίσης, αποστέλλει στις συσκευές δυναμικές ρυθμίσεις (μέσω ADR) για την προσαρμογή του ρυθμού μετάδοσης και της ισχύος, βελτιώνοντας την αξιοπιστία και μειώνοντας την κατανάλωση ισχύος των κόμβων [37].

**Application Server (AS):** Διαχειρίζεται το ανώτερο επίπεδο εφαρμογής του LoRaWAN.

Συγκεκριμένα, αναλαμβάνει την αποκρυπτογράφηση των φορτίων δεδομένων (payloads) που προέρχονται από τις συσκευές και την παράδοσή τους στις αντίστοιχες εφαρμογές, καθώς και την κρυπτογράφηση των δεδομένων που στέλνονται προς τις συσκευές (downlink). Ο Application Server μπορεί επίσης να περνάει τα δεδομένα από μορφοποιητές ωφέλιμου φορτίου (payload formatters) για να μετασχηματίζει τα δυαδικά δεδομένα των αισθητήρων σε ευανάγνωστη μορφή (π.χ. βαθμοί Κελσίου, ποσοστά, κ.λπ.) πριν τα παραδώσει στις εφαρμογές. Ένα σημαντικό χαρακτηριστικό είναι ότι ο AS επιτρέπει την ένταξη των δεδομένων σε εξωτερικές πλατφόρμες ή υπηρεσίες cloud. Για παράδειγμα, υποστηρίζονται διασυνδέσεις με δημοφιλείς IoT πλατφόρμες (AWS IoT, Azure IoT, Google Cloud κ.ά.), αποθήκευση των μηνυμάτων σε βάσεις δεδομένων ή διοχέτευση των δεδομένων μέσω δικτυακών APIs προς τρίτα συστήματα. Στο πλαίσιο ενός ιδιωτικού συστήματος, η συνηθέστερη μέθοδος είναι η χρήση του πρωτοκόλλου MQTT ή/και HTTP για την παράδοση των δεδομένων σε εφαρμογές σε πραγματικό χρόνο [39].

**Join Server (JS):** Είναι υπεύθυνος για τη διεκπεραίωση της διαδικασίας ενεργοποίησης των συσκευών στο δίκτυο (Join procedure του LoRaWAN). Συγκεκριμένα, ο Join Server φυλάσσει τα μοναδικά κλειδιά που αντιστοιχούν σε κάθε συσκευή (π.χ. το AppKey για OTAA) και συμμετέχει στην ανταλλαγή μηνυμάτων join-request/join-accept με τη συσκευή. Μετά την επιτυχή ενεργοποίηση μιας συσκευής, ο JS παράγει και διανέμει τα κλειδιά συνεδρίας (session keys) τόσο στον Network Server όσο και στον Application Server, ώστε να μπορούν να πραγματοποιούν την κρυπτογράφηση και αυθεντικοποίηση των μηνυμάτων κατά τη διάρκεια της λειτουργίας [33]. Στο οικοσύστημα του The Things Stack, ο Join Server μπορεί να λειτουργεί και ως παγκόσμιος εξυπηρετητής (Global Join Server) κοινός για πολλαπλά δίκτυα, διευκολύνοντας την περιαγωγή συσκευών μεταξύ δικτύων [39].

**Identity Server (IS):** Αποτελεί τον μηχανισμό διαχείρισης ταυτοτήτων και δικαιωμάτων στο σύστημα. Διατηρεί μητρώα όλων των οντοτήτων του δικτύου: χρήστες και οργανισμοί (για την πολυ-ενοικιαστική υποστήριξη), εφαρμογές, συσκευές και gateways, καθώς και πελάτες OAuth και παρόχους ελέγχου πρόσθασης. Μέσω του IS γίνεται ο έλεγχος προσθάσεων με χρήση API keys και ρόλων. Η ύπαρξη ξεχωριστού Identity Server επιτρέπει στο TTS να υποστηρίζει πολλαπλούς ταυτόχρονους χρήστες/πελάτες σε ένα κοινό δίκτυο με ασφαλή και απομονωμένο τρόπο, κρίσιμο χαρακτηριστικό για μεγάλες εγκαταστάσεις με πολλούς οργανισμούς, αλλά χρήσιμο ακόμη και σε μικρότερα ιδιωτικά δίκτυα [39].

**Console:** Παρόλο που δεν αποτελεί ανεξάρτητη υπηρεσία του backend, αξίζει να αναφερθεί το Console, η διαδικτυακή εφαρμογή διεπαφής χρήστη του The Things Stack. Πρόκειται για έναν διαδικτυακό πίνακα ελέγχου μέσω του οποίου ο διαχειριστής ή χρήστης του δικτύου μπορεί εύκολα να προσθαφαιρεί συσκευές και gateways, να παρακολουθεί την κατάσταση τους και την κίνηση των δεδομένων σε πραγματικό χρόνο και να ρυθμίζει παραμέτρους του δικτύου μέσω μιας φιλικής γραφικής διεπαφής χρήστη/ Graphical User Interface (GUI). Η Console επικοινωνεί με τις παρασκηνιακές υπηρεσίες μέσω των API που προσφέρει το TTS. Εναλλακτικά, για αυτοματοποιημένη διαχείριση και προχωρημένες ρυθμίσεις, το TTS προσφέρει και γραμμή εντολών (CLI) καθώς και απευθείας REST/gRPC API ώστε οι προγραμματιστές να ενσωματώνουν λειτουργίες του TTS στις δικές τους εφαρμογές [39].

**Άλλες βιβλιοθήκες/τεχνολογίες:** Εν κατακλείδι, οισμένες ακόμη τεχνολογίες χρήζουν νύξης. Για την αποθήκευση των δεδομένων του Network Server (π.χ. στοιχεία συσκευών, χρήστες κ.ά.) χρησιμοποιείται μία βάση δεδομένων PostgreSQL, σύμφωνα με τις απαιτήσεις του TTS. Επιπλέον, το TTS αξιοποιεί την in-memory βάση Redis ως cache για γρήγορη αποθήκευση προσωρινών δεδομένων (όπως sessions συσκευών, μετρικές ρυθμού μηνυμάτων, κ.λπ.) μειώνοντας το φορτίο στη βάση δεδομένων PostgreSQL και επιταχύνοντας τις λειτουργίες. Η Redis είναι εξαιρετικά ταχεία σε αναγνώσεις/εγγραφές και χρησιμοποιεί δομές δεδομένων στη μνήμη, ιδανική για τέτοια χρήση. Ακόμη, στη στοίβα μας περιλαμβάνεται ο MQTT client library για Python ή Node-RED nodes, στην περίπτωση που θέλουμε να υλοποιήσουμε λογική στην άκρη της εφαρμογής (π.χ. ένας Node-RED flow που λαμβάνει MQTT μηνύματα και στέλνει ειδοποίηση). Το συνοθύλευμα των ανωτέρω συνθέτουν μια συνεκτική πλατφόρμα λογισμικού όπου κάθε στοιχείο έχει σαφή ρόλο, από το επίπεδο του υλικού (αισθητήρες, gateways) και τα δίκτυα επικοινωνίας (LoRa) μέχρι το επίπεδο μεταφοράς δεδομένων (MQTT/HTTP) και τελικά την παρουσίαση ή αποθήκευση των μετρήσεων.

Στο πλαίσιο της παρούσας εργασίας, χρησιμοποιήθηκε η ανοικτή έκδοση του The Things Stack ως LoRaWAN Network Server. Η εγκατάστασή του πραγματοποιήθηκε σε περιβάλλον Docker (βλ. ενότητα 3.3), όπου κάθε υπηρεσία του TTS εκτελείται σε ξεχωριστό κοντέινερ (container). Έγινε κατάλληλη ρύθμιση των παραμέτρων του Identity Server και του Application Server ώστε να συνδέονται με τη βάση δεδομένων και να εξυπηρετούν τις απαιτήσεις του δικτύου. Επιπλέον, δημιουργήθηκε μέσω της διεπαφής διαχείρισης (Console) μια καταχώρηση για το gateway του συστήματος και εκδόθηκε ένα κλειδί API για την πιστοποίηση του gateway κατά τη σύνδεσή του. Το gateway συνδέεται ασφαλώς με το TTS μέσω του Gateway Server, χρησιμοποιώντας το πρωτόκολλο LoRa Basics Station, όπως περιγράφεται στη συνέχεια.

Τέλος, για την ενσωμάτωση των δεδομένων αισθητήρων στην εφαρμογή, το TTS παρέχει μηχανισμό webhook. Συγκεκριμένα, χρησιμοποιήθηκε μια διεπαφή HTTP Webhook Integration του The Things Stack, μέσω της οποίας κάθε νέο uplink μήνυμα που λαμβάνει ο Application Server προωθείται αυτόματα (μέσω HTTP POST) στην εφαρμογή Spring Boot του συστήματος. Κατ' αυτόν τον τρόπο, επιτυγχάνεται σε πραγματικό χρόνο η μεταφορά των μετρήσεων από το δίκτυο LoRaWAN προς τον διακομιστή της εφαρμογής. Η επιλογή των webhooks καθιστά την ενσωμάτωση (integration) απλούστερη, αξιοποιώντας απευθείας κλήσεις HTTP προς το backend, όπως ήταν επιθυμητό στην αρχιτεκτονική της εφαρμογής.

## Πλεονεκτήματα

To The Things Stack έχει καθιερωθεί ως μία από τις πλέον ολοκληρωμένες λύσεις LoRaWAN διακομιστή, με ενεργή κοινότητα και υποστήριξη από τον οργανισμό The Things Network. Ως ανοιχτού κώδικα λογισμικό, προσφέρει διαφάνεια στον τρόπο λειτουργίας και δυνατότητα προσαρμογής ή επέκτασης από την κοινότητα. Επιπλέον, υποστηρίζει το πλήρες εύρος του προτύπου LoRaWAN (έως έκδοση 1.0.4/1.1) όσον αφορά τις κλάσεις συσκευών, τις διαδικασίες αυθεντικοποίησης και την κρυπτογραφία, εξασφαλίζοντας ότι οι συσκευές που συμμορφώνονται με το πρότυπο θα μπορούν να επικοινωνούν απρόσκοπτα. Η αρχιτεκτονική microservices του επιτρέπει εύκολη οριζόντια κλιμάκωση (π.χ. μπορούν να τρέχουν

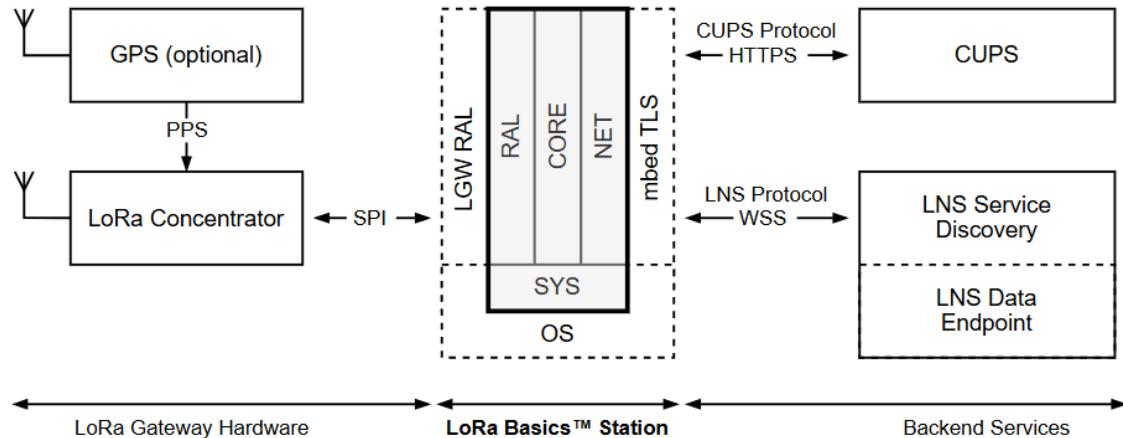
πολλαπλές εμφανίσεις του NS ή του AS πίσω από τον load balancer για υψηλότερη χωρητικότητα). Επίσης, το TTS παρέχει εργαλεία ανάπτυξης όπως τη γραμμή εντολών ttn-lw-cli και εκτενές API, που διευκολύνουν την αυτοματοποίηση στη διαχείριση μεγάλων στόλων συσκευών. Τέλος, η κοινότητα TTN προσφέρει public servers (Community network) όπου μπορεί κανείς να συνδέσει συσκευές δωρεάν, γεγονός που αποτελεί ένδειξη ωριμότητας και αξιοπιστίας της στοίβας αυτής, καθώς υποστηρίζει εκατομμύρια μηνύματα καθημερινά σε όλο τον κόσμο [39].

## Προκλήσεις / Περιορισμοί

Παρόλο που η open-source έκδοση παρέχει τις βασικές λειτουργίες, δεν περιλαμβάνει ορισμένα προηγμένα χαρακτηριστικά που διατίθενται στις εμπορικές εκδόσεις (π.χ. ολοκληρωμένα εργαλεία διαχείρισης πολλών tenants, ενσωματωμένο Packet Broker peering, κ.ά.). Αυτό συνεπάγεται ότι για ιδιαίτερα μεγάλες εγκαταστάσεις ενδέχεται να απαιτείται αναβάθμιση ή αυξημένος βαθμός χειροκίνητης παρέμβασης. Η αρχική εκμάθηση και παραμετροποίηση του TTS μπορεί να είναι απαιτητική, καθώς εμπλέκει πολλά στοιχεία (components) και ρυθμίσεις (π.χ. περιφερειακές παράμετροι, διαχείριση κλειδιών, certificates για TLS κ.λπ.). Επιπλέον, η εγκατάσταση της πλατφόρμας σε on-premises περιβάλλον συνεπάγεται την ανάγκη συντήρησης των υποδομών (βάσεις δεδομένων, ενημερώσεις λογισμικού, παρακολούθηση απόδοσης). Τέλος, επειδή το LoRaWAN εξελίσσεται, η συμβατότητα με μελλοντικές εκδόσεις του προτύπου απαιτεί αναβάθμιση του ίδιου του TTS (η κοινότητα παρέχει συχνές ενημερώσεις), αλλά ο ρυθμός αυτός πρέπει να παρακολουθείται από τον διαχειριστή του συστήματος.

## 3.2 LoRa Basics™ Station

To LoRa Basics™ Station αποτελεί λογισμικό gateway που αναπτύχθηκε από Semtech και αντιπροσωπεύει τη νέα γενιά προωθητή πακέτων (packet forwarder) για δίκτυα LoRaWAN. Βασίζεται σε σύγχρονες αρχές σχεδίασης για να αντικαταστήσει τον παλαιότερο απλούστερο UDP packet forwarder, προσφέροντας αυξημένη ασφάλεια και δυνατότητες διαχείρισης σε μεγάλη κλίμακα. Το LoRa Basics Station τρέχει τοπικά στο hardware του LoRaWAN gateway (π.χ. σε ένα Linux-based board που φέρει τον συγκεντρωτή SX1301/SX1302) και είναι υπεύθυνο να γεφυρώσει τον ασύρματο κόσμο των συσκευών LoRa με το IP δίκτυο προς τον Network Server. Συγκεκριμένα, λαμβάνει μέσω του RF frontend του gateway όλα τα πακέτα LoRaWAN (uplinks) που εκπέμπουν οι τερματικές συσκευές και τα προωθεί μέσω διαδικτύου προς τον LoRaWAN Network Server (όπως το TTS), ενώ αντίστροφα λαμβάνει από τον Network Server πακέτα για downlink και τα μεταδίδει από το ραδιόφωνο του gateway προς τις συσκευές στους κατάλληλους χρόνους. Με αυτόν τον τρόπο, το LoRa Basics Station λειτουργεί ως ο τοπικός αντιπρόσωπος του δικτύου στο σημείο του gateway, χειριζόμενο όλες τις λεπτομέρειες του radio protocol [40, 41].



Εικόνα 3.2: Αρχιτεκτονική του LoRa Basics™ Station.

[40]

### Πρωτόκολλα LNS και CUPS

Σε αντίθεση με τον απλό UDP forwarder, το LoRa Basics Station υιοθετεί δύο παράλληλα πρωτόκολλα επικοινωνίας πάνω από IP (τυπικά TCP/IP με TLS), το LNS protocol και το CUPS protocol. Το πρωτόκολλο LNS (LoRaWAN Network Server protocol) αφορά τη ζωντανή μεταφορά δεδομένων μεταξύ gateway και δικτύου. Υλοποιείται συνήθως ως μια επικοινωνία WebSockets (secure WS) με τον Network Server, όπου το gateway ανοίγει μία μόνιμη σύνδεση προς το URL του LNS (π.χ. `wss://<ns-address>:8887`) μέσω της οποίας αποστέλλει τα uplinks (ως JSON μηνύματα) και λαμβάνει downlinks ή εντολές από τον server. Όλα τα μηνύματα LNS είναι κρυπτογραφημένα με TLS και το LoRa Basics Station υποστηρίζει αυθεντικοποίηση είτε με πιστοποιητικά TLS είτε με token που έχει εκδοθεί από τον Network Server για το συγκεκριμένο gateway (η δεύτερη μέθοδος θεωρείται πιο ελαφριά και κατάλληλη για gateways χαμηλών πόρων). Το δεύτερο πρωτόκολλο, το CUPS (Configuration and Update Server protocol) χρησιμοποιεί ξεχωριστή περιοδική επικοινωνία του gateway με έναν ειδικό διακομιστή (CUPS server) για κεντρική διαχείριση και ενημερώσεις. Μέσω CUPS, ένα gateway μπορεί επί παραδείγματι να λαμβάνει ενημερώσεις λογισμικού (firmware) ή νέες ρυθμίσεις διαμόρφωσης (συχνότητες, power, κλειδιά) από απόσταση, χωρίς ανθρώπινη παρέμβαση. Το LoRa Basics Station συνδέεται ανά τακτά διαστήματα (ή κατ' απαίτηση) με τον CUPS server μέσω HTTPS, ελέγχει αν υπάρχει διαθέσιμη νέα διαμόρφωση ή αναβάθμιση και την εφαρμόζει με ασφαλή τρόπο, ενώ υποστηρίζονται ψηφιακές υπογραφές ECDSA για την ακεραιότητα των ενημερώσεων. Τα δύο αυτά πρωτόκολλα επιτρέπουν την κεντρικοποιημένη διαχείριση στόλου gateways, δηλαδή ο διαχειριστής του δικτύου μπορεί από ένα σημείο να στέλνει ενημερώσεις και να λαμβάνει στατιστικά από εκατοντάδες gateways, χωρίς να απαιτείται τοπική πρόσθαση στο καθένα [40, 41].

### Χαρακτηριστικά και λειτουργία

Το LoRa Basics Station έχει σχεδιαστεί για να υποστηρίζει πλήρως τις ιδιαιτερότητες του LoRaWAN πρωτοκόλλου και των κλάσεων επικοινωνίας. Υποστηρίζει επικοινωνία Class A (διεπαφή χωρίς συνεχή ακρόαση), Class C (συνεχής ακρόαση για άμεσα downlinks) αλλά και

Class B (προγραμματισμένα beaconing downlinks), αξιοποιώντας τεχνικές συγχρονισμού χρόνου μέσω GPS ή μέσω χρονικών σημάτων από τον server. Μάλιστα, για Class B το LoRa Basics Station μπορεί να λειτουργήσει χωρίς το gateway να έχει GPS, λαμβάνοντας συγχρονισμό χρόνου από τον ίδιο τον Network Server και καθοδηγώντας το gateway να εκπέμψει ειδικά beacon σήματα με σωστό συγχρονισμό (λειτουργία Server Assisted GPS-less Beaconing). Ένα άλλο καινοτόμο χαρακτηριστικό είναι ότι το LoRa Basics Station δεν απαιτεί καμία εισερχόμενη σύνδεση από το δίκτυο προς το gateway, μιας και όλη η επικοινωνία γίνεται με εξερχόμενες συνδέσεις που ξεκινά το ίδιο το gateway (για LNS και CUPS). Αυτό καθιστά τη λειτουργία του firewall-friendly, δηλαδή μπορεί να δουλέψει πίσω από NAT/Firewall χωρίς περίπλοκες ρυθμίσεις (σε αντίθεση με παλαιότερες προσεγγίσεις που απαιτούσαν ανοικτές πόρτες για downlink push από τον server). Επίσης, το LoRa Basics Station παρέχει δυνατότητα απομακρυσμένου shell, δηλαδή, μέσω της ασφαλούς σύνδεσης LNS, μπορεί ο Network Server (εφόσον επιτραπεί) να ανοίξει ένα secure shell session προς το gateway για σκοπούς debugging, καταργώντας την ανάγκη για ξεχωριστά τούνελ (tunnels) SSH [41].

Στο εσωτερικό του, το LoRa Basics Station είναι γραμμένο σε γλώσσα προγραμματισμού C και σχεδιασμένο να είναι φορητό και αποδοτικό. Η αρχιτεκτονική του χωρίζεται σε ένα portability layer (RAL, HAL modules) που προσαρμόζεται ανάλογα με το hardware του gateway και σε έναν πυρήνα ανεξάρτητο από πλατφόρμα που υλοποιεί τη λογική πολυδιεργασίας, τη διαχείριση πακέτων και τα πρωτόκολλα. Αυτό κάνει σχετικά εύκολη τη μεταφορά (porting) του LoRa Basics Station σε νέα μοντέλα gateways ή και σε ενοωματωμένα συστήματα με περιορισμένους πόρους. Ήδη από το 2020, μεγάλοι κατασκευαστές gateway (Laird, RAK, Brown κ.λπ.) έχουν υιοθετήσει το LoRa Basics Station στα προϊόντα τους και η τάση αυτή ενισχύεται. Η The Things Network έχει καθιερώσει το LoRa Basics Station ως τον προτεινόμενο τρόπο σύνδεσης gateway στο δίκτυο της, αντικαθιστώντας σταδιακά τον UDP forwarder χάρις τα πλεονεκτήματα που προσφέρει ως προς την ασφάλεια και τη διαχείριση [41].

## Συμβατότητα με The Things Stack

Το TTS (ιδίως η έκδοση v3) υποστηρίζει πλήρως το πρωτόκολλο LNS του Basics Station. Όταν καταχωρείται ένα gateway στο TTS, αυτό παράγει τα απαραίτητα διαπιστευτήρια (π.χ. ένα LNS API key ή/και client certificate) και παρέχει τη διεύθυνση LNS (URI) η οποία απαιτεί ρύθμιση εντός του λογισμικού Basics Station του gateway. Επιπλέον, το TTS περιλαμβάνει ειδική υπηρεσία Gateway Configuration Server (GCS) που μπορεί να λειτουργήσει ως CUPS server για όσα gateways το υποστηρίζουν. Έτσι, σε ένα ιδιωτικό δίκτυο, ο διαχειριστής μπορεί να χρησιμοποιήσει το ίδιο το TTS για να αποστέλλει ενημερώσεις ρυθμίσεων στα gateway (π.χ. αλλαγές στο channel plan) μέσω CUPS ή εναλλακτικά να αξιοποιήσει τον GCS για να δημιουργήσει έτοιμα configuration files για gateways που χρησιμοποιούν UDP (σε legacy περιπτώσεις) [40, 41].

## Πλεονεκτήματα

To LoRa Basics Station προσφέρει υψηλότερο επίπεδο ασφάλειας σε σχέση με τον πρόκατοχό του, καθώς όλες οι επικοινωνίες γίνονται μέσω TLS με έλεγχο ταυτότητας, εξαλείφοντας τις ευπάθειες της ανεξέλεγκτης UDP μετάδοσης. Επιτρέπει την αποτελεσματική κεντρική διαχείριση μεγάλου αριθμού gateways, καθώς οι διαχειριστές μπορούν να αναβαθμίζουν το λογισμικό και να αλλάζουν ρυθμίσεις απομακρυσμένα μειώνοντας το λειτουργικό κόστος μεγάλων δικτύων. Η αρχιτεκτονική του είναι επεκτάσιμη και προσαρμόσιμη σε διαφορετικά λειτουργικά και hardware, διασφαλίζοντας μελλοντική συμβατότητα καθώς το οικοσύστημα LoRaWAN εξελίσσεται. Επιπλέον, προσφέρει βελτιστοποιήσεις για αξιόπιστη λειτουργία, όπως για παραδειγματικό το μηχανισμό της συγχρονισμού χρόνου για Class B, ανθεκτικότητα σε διακοπές δικτύου (buffering πακέτων) και λεπτομερή αναφορά κατάστασης gateway προς τον server. Τέλος, είναι ανοιχτού κώδικα (διαθέσιμο στο GitHub της Semtech) και έχει ευρεία αποδοχή, κάτι που σημαίνει ότι υπόκειται σε συνεχείς βελτιώσεις και ελέγχους από την κοινότητα.

## Προκλήσεις

Η μετάβαση από τον παλαιό UDP forwarder στο LoRa Basics Station ενδέχεται να απαιτήσει πρόσθετη πολυπλοκότητα στην αρχική ρύθμιση. Συγκεκριμένα, πρέπει να δημιουργηθούν και να διαχειρίζονται πιστοποιητικά TLS ή κλειδιά token για κάθε gateway, ειδάλλως το Basics Station δεν θα συνδεθεί. Επίσης, μερικά παλαιότερα μοντέλα gateway μπορεί να μην υποστηρίζουν επίσημα το Station firmware, οπότε ίσως απαιτείται αναβάθμιση λογισμικού από τον κατασκευαστή. Ακόμη, σε περίπτωση προσωρινής απώλειας συνδεσιμότητας στο internet το Basics Station διατηρεί τα πακέτα σε buffer, αλλά αν η διακοπή παραταθεί υπάρχει κίνδυνος απώλειας δεδομένων (όπως συμβαίνει γενικά με uplinks που δεν παραδόθηκαν). Λαμβάνοντας υπόψιν το σύνολο των παραμέτρων, τα οφέλη υπερτερούν των μειωνεκτημάτων, με αποτέλεσμα, για επαγγελματικές εγκαταστάσεις μεγάλης κλίμακας, το Basics Station θεωρείται πλέον μονόδρομος για ασφαλή και αποδοτική διαχείριση των gateways.

## 3.3 Docker

Η πλατφόρμα Docker αποτελεί μια πλατφόρμα ανοιχτού κώδικα που επιτρέπει την αυτοματοποίηση της ανάπτυξης, διανομής και εκτέλεσης εφαρμογών μέσα σε ελαφριά απομονωμένα περιβάλλοντα που ονομάζονται containers. Η αξιοποίηση του Docker παρέχει τη δυνατότητα ενσωμάτωσης μιας εφαρμογής (μαζί με όλες τις απαιτούμενες εξαρτήσεις της) σε μία αυτοτελή εικόνα λογισμικού (image), διασφαλίζοντας ότι η εφαρμογή θα εκτελείται με ομοιομορφία σε οποιοδήποτε σύστημα φιλοξενίας (host) ανεξάρτητα από την αρχιτεκτονική και το λογισμικό του συστήματος ή των εγκατεστημένων βιτσλιοθηκών. Το container περιλαμβάνει ότι χρειάζεται η εφαρμογή και απομονώνει το περιβάλλον εκτέλεσής (runtime) της από το λειτουργικό σύστημα (Operating System, OS) του host. Στο πλαίσιο του LoRaWAN συστήματος μας, το Docker χρησιμοποιείται για να φιλοξενήσει υπηρεσίες όπως το The Things Stack και τα συναφή υποσυστήματά του, διευκολύνοντας την εγκατάσταση και τη συντήρησή τους [42].

## Αρχιτεκτονική Docker (Client-Server)

Η λειτουργία του Docker ακολουθεί αρχιτεκτονική πελάτη-εξυπηρετητή (client-server). Υπάρχει ένας Docker daemon (γνωστός και ως dockerd) που τρέχει στο host σύστημα ως υπηρεσία και δέχεται εντολές μέσω ενός Docker client. Ο Docker client είναι συνήθως η εντολή γραμμής docker με την οποία ο χρήστης εκτελεί εντολές όπως docker run, docker build, docker stop κ.λπ. Ο client αυτός στέλνει τις εντολές στο daemon μέσω ενός REST API (υλογοιημένο πάνω σε UNIX socket ή μέσω network interface). Ο Docker daemon αναλαμβάνει το μεγάλο φόρτο εργασίας: κατασκευή εικόνων, εκτέλεση και διαχείριση containers, διαχείριση του συστήματος αρχείων των containers, δικτύων, volumes κ.ά. Ο client και ο daemon μπορούν να τρέχουν στο ίδιο μηχάνημα (τυπικά σε μια «μονοκόμματη» ανάπτυξη) ή ο client να ελέγχει ένα απομακρυσμένο Docker host μέσω δικτύου.

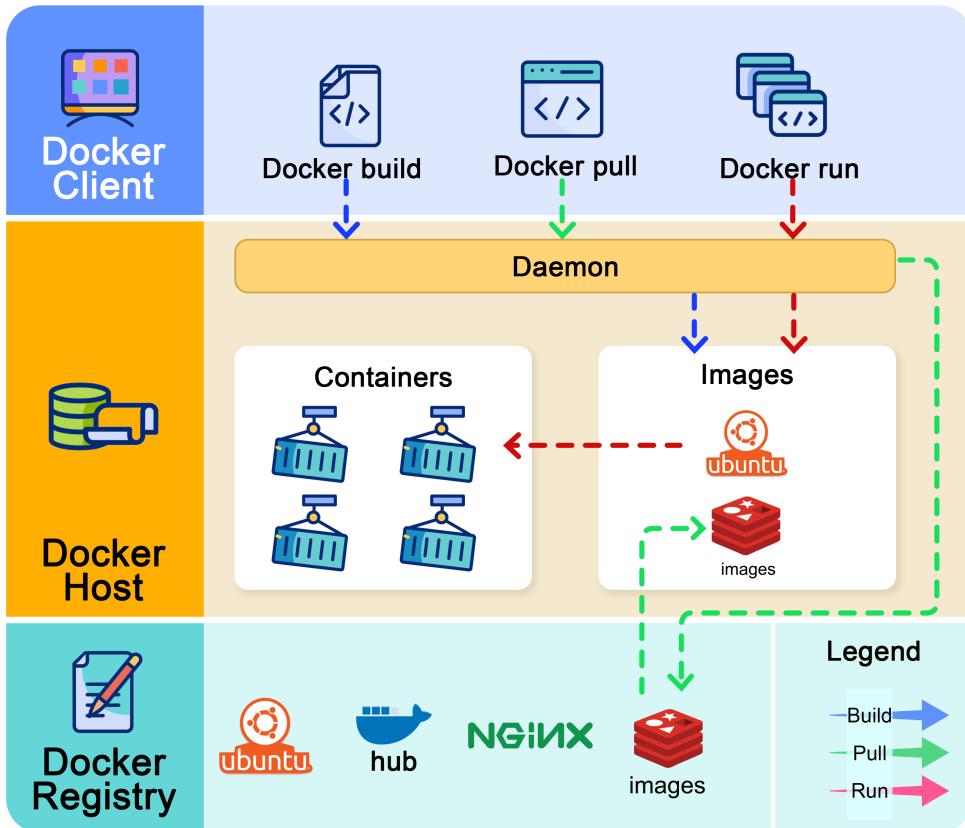
Στο Docker περιλαμβάνεται επίσης η έννοια του Docker Registry, ενός αποθετηρίου (με γνωστότερο το Docker Hub) όπου αποθηκεύονται και διανέμονται οι εικόνες λογισμικού. Όταν εκτελούμε docker pull ή docker run για μια εικόνα, ο Docker daemon ανακτά την εικόνα από το αποθετήριο (registry) (αν δεν υπάρχει τοπικά), ενώ με την εντολή docker push μπορούμε να «ανεβάσουμε» δικές μας εικόνες στο registry. Το Docker image είναι ένα πακέτο που περιλαμβάνει όλα τα στρώματα λογισμικού που απαιτούνται για ένα container (συνήθως βασίζεται σε μια ελαφριά διανομή Linux και περιλαμβάνει τις απαραίτητες βιβλιοθήκες και το εκτελέσιμο αρχείο της εφαρμογής μας). Τα images είναι διαστρωματωμένα, δηλαδή κάθε αλλαγή/εντολή στο Dockerfile δημιουργεί ένα νέο layer, επιτρέποντας την επαναχρησιμοποίηση κοινών στρωμάτων μεταξύ εικόνων ώστε να είναι πιο ελαφριές και γρήγορες [42, 43].

## Docker container

Ένα Docker container είναι το εκτελέσιμο στιγμιότυπο (instance) μιας εικόνας (μπορούμε να το παρομοιάσουμε με τη δημιουργία μιας διεργασίας από ένα διαδικό (binary) εκτελέσιμο αρχείο στο λειτουργικό σύστημα). Το container τρέχει απομονωμένα, αξιοποιώντας λειτουργίες του πυρήνα Linux (Linux kernel) όπως χώροι ονομάτων πυρήνα (kernel namespaces) και cgroups (control groups) για να διασφαλίσει ότι το σύστημα αρχείων (filesystem), η δικτύωση και οι πόροι του container είναι ξεχωριστοί από του host. Η απομόνωση αυτή είναι σχεδόν πλήρης, αλλά με πολύ μικρή επιβάρυνση (overhead) συγκριτικά με μια πλήρη εικονική μηχανή (virtual machine), καθώς όλα τα containers μοιράζονται τον ίδιο πυρήνα του host λειτουργικού. Έτσι, το Docker παρέχει μια λύση ελαφριάς εικονικοποίησης, αφού έχουμε παρόμοια οφέλη με των VMs (απομονωμένο περιβάλλον, σταθερό runtime), αλλά χωρίς την επιβάρυνση να εκτελείται ξεχωριστός πυρήνας και λειτουργικό σύστημα για κάθε instance [42, 43].

## Docker compose

Το Docker από μόνο του διαχειρίζεται μεμονωμένα containers. Στην πράξη όμως, μια σύνθετη εφαρμογή (όπως το σύστημά μας) αποτελείται από πολλαπλές υπηρεσίες που τρέχουν συνεργατικά σε διαφορετικά containers (π.χ. ένα container για τον The Things



Εικόνα 3.3: Αρχιτεκτονική Docker: ο Docker Client εκτελεί εντολές (build, pull, run) προς τον Docker daemon, που δημιουργεί/χειρίζεται images και containers και επικοινωνεί με τον Docker Registry.

[44]

Stack server, ένα για τη βάση δεδομένων του, ένα για έναν MQTT broker κ.ο.κ). Το Docker compose αποτελεί ένα εργαλείο (έναν ειδικό Docker client) που επιτρέπει τον πλήρη ορισμό όλων των container σε ένα αρχείο YAML όλα τα containers μιας πολυ-υπηρεσιακής εφαρμογής καθώς και των μεταξύ τους συνδέσεων όπως δίκτυα, volumes και μεταβλητές περιβάλλοντος. Με μια απλή εντολή docker-compose up μπορούμε να εκκινήσουμε το σύνολο των υπηρεσιών στη σωστή σειρά και με docker-compose down να προσθούμε σε μαζικό τερματισμό [42, 43].

Στην παρούσα εφαρμογή, το Docker compose χρησιμοποιείται για να δημιουργηθεί ένα περιβάλλον που περιλαμβάνει όλα τα επιμέρους κομμάτια του LoRaWAN network server (TTS) μαζί με τις εξαρτήσεις του, κάτι που αυτοματοποιεί σε τεράστιο βαθμό τη διαδικασία εγκατάστασης. Επί παραδείγματι, η ανοιχτού κώδικα έκδοση του TTS διαθέτει ένα έτοιμο αρχείο docker-compose.yml που ορίζει υπηρεσίες για το Stack, το PostgreSQL, το Redis, κ.λπ., ώστε ο χρήστης να μπορεί με μία κίνηση να ανεβάσει ολόκληρη τη στοίβα. Επιπλέον, ακολουθώντας την ίδια αρχή στήνονται containers για το LoRa Basics Station καθώς και για το backend και το frontend της εφαρμογής χρήστη για την απεικόνιση των δεδομένων. Το compose χειρίζεται επίσης το networking (δημιουργώντας ένα ιδιωτικό δίκτυο docker όπου επικοινωνούν τα containers μεταξύ τους) και την αποθήκευση (π.χ. δύναται να δημιουργεί volumes ώστε η βάση δεδομένων να αποθηκεύει δεδομένα μόνιμα στο host).

## Πλεονεκτήματα

Η χρήση του Docker απλοποίησε σημαντικά την ανάπτυξη και διαχείριση εφαρμογών. Χωρίς αυτό, ο εγκαταστάτης ενός LoRaWAN network server θα έπρεπε να ρυθμίσει χειροκίνητα γλώσσες προγραμματισμού, βάσεις δεδομένων και βιβλιοθήκες, διαδικασία χρονοβόρα και επιρρεπή σε σφάλματα. Με το Docker, όλα «πακετάρονται» σε images που μπορούν να εκτελεστούν σε οποιοδήποτε σύστημα, μειώνοντας το κόστος και την πολυπλοκότητα. Διευκολύνονται επίσης οι ενημερώσεις, καθώς μια υπηρεσία μπορεί να αναβαθμιστεί απλά με νέο image, χωρίς να επηρεαστούν οι υπόλοιπες. Η απομόνωση των containers προλαμβάνει συγκρούσεις ρυθμίσεων και προάγει τη φορητότητα, αφού το ίδιο container μπορεί να λειτουργήσει τοπικά ή σε απομακρυσμένο server με την ίδια συμπεριφορά.

## Προκλήσεις

Παρά τα οφέλη, τα containers εισάγουν επιπλέον πολυπλοκότητα, καθώς απαιτούν εξουκείωση με έννοιες όπως images, volumes και container networking. Η εκσφαλμάτωση ενδέχεται να είναι πιο απαιτητική και η απομόνωση δεν εξασφαλίζει απόλυτη ασφάλεια, ιδιαίτερα σε πειπτώσεις όπου κάποιο container τρέχει με δικαιώματα root. Σε μεγάλες εγκαταστάσεις χρειάζονται εργαλεία ορχήστρωσης όπως το Kubernetes, ωστόσο για μεσαίας κλίμακας συστήματα το Docker compose παρέχει επαρκείς δυνατότητες οργάνωσης.

## 3.4 Spring Boot

Το Spring Boot είναι ένα σύγχρονο πλαίσιο ανάπτυξης εφαρμογών Java που διευκολύνει τη δημιουργία αυτόνομων, παραγωγικών web εφαρμογών με ελάχιστες απαιτούμενες ρυθμίσεις. Αποτελεί επέκταση του οικοσυστήματος Spring Framework παρέχοντας προκαθορισμένες διαμορφώσεις (auto-configurations) και έτοιμες ενσωματώσεις για πολλά συνήθη στοιχεία μίας εφαρμογής (όπως web server, ασφάλεια, πρόσθαση σε βάση δεδομένων κ.ά.). Βασικός στόχος του Spring Boot είναι το «συμφωνημένο αντί της ρύθμισης» (convention over configuration), τουτέστιν προσφέρει λογικές προεπιλεγμένες ρυθμίσεις ώστε ο προγραμματιστής να μπορεί να εκκινήσει άμεσα την εφαρμογή του χωρίς να ασχοληθεί με λεπτομερείς ρυθμίσεις αρχείων. Για παράδειγμα, μια εφαρμογή Spring Boot περιλαμβάνει ενσωματωμένο διακομιστή HTTP (όπως Tomcat ή Jetty), ο οποίος εκκινεί αυτόματα, επιτρέποντας στην εφαρμογή να τρέξει ως αυτόνομο Java jar αρχείο [45, 46].

To Spring Boot υλοποιεί το πρότυπο MVC (Model-View-Controller) για την κατασκευή web applications. Παρέχει μηχανισμούς για τη δημιουργία RESTful APIs πολύ εύκολα, μέσω «σημειώσεων» (annotations) π.χ. @RestController και @RequestMapping. Επιπλέον, ενσωματώνεται άριστα με βιβλιοθήκες όπως το Spring Data JPA (Java Persistence API) για την αλληλεπίδραση με σχεσιακές βάσεις δεδομένων, προσφέροντας αφαιρετικό επίπεδο πάνω από το Object-relational mapping (ORM) π.χ. Hibernate για τη διαχείριση οντοτήτων κερδίζοντας σε παραγωγικότητα. Ενσωματωμένες επίσης είναι λύσεις για το logging, το monitoring (μέσω Actuator), καθώς και για την ασφαλή διάθεση APIs (Spring Security), χωρίς ο προγραμματιστής να χρειάζεται να ξεκινήσει εκ βάθρων [45, 46].

Στην υλοποίηση του παρόντος συστήματος, το Spring Boot χρησιμοποιήθηκε για την ανάπτυξη του κεντρικού διακομιστή εφαρμογής (backend). Η εφαρμογή αυτή αναλαμβάνει πολλούς κρίσιμους ρόλους.

Αρχικά, λειτουργεί ως σημείο λήψης δεδομένων από το δίκτυο LoRaWAN. Όπως περιγράφηκε, μέσω της ενσωματωσης webhook του TTS, κάθε φορά που μια συσκευή στέλνει μια μέτρηση, ο Application Server του TTS προωθεί τα δεδομένα με ένα αίτημα HTTP προς την εφαρμογή Spring Boot. Στην πλευρά της εφαρμογής, έχει υλοποιηθεί ένας REST controller ο οποίος παρέχει το κατάλληλο endpoint (π.χ. ένα URL όπου δέχεται POST requests) για να δέχεται αυτά τα εισερχόμενα δεδομένα. Στη συνέχεια, τα δεδομένα αποθηκεύονται στη βάση PostgreSQL για μετέπειτα χρήση και ανάλυση.

Πέρα από τη λήψη και αποθήκευση δεδομένων, το Spring Boot backend παρέχει και μια σειρά από υπηρεσίες προς το frontend (την εφαρμογή σε React JS, βλ. Ενότητα 3.5). Έχουν υλοποιηθεί διάφορα API endpoints (π.χ. τύπου REST GET) τα οποία επιτρέπουν στην διεπαφή χρήστη να αντλεί πληροφορίες από τη βάση. Ενδεικτικά, υπάρχει endpoint που επιστρέφει το ιστορικό των μετρήσεων αισθητήρων ή την πιο πρόσφατη ένδειξη και την κατάστασή της. Με αυτόν τον τρόπο, το frontend μπορεί να παρουσιάζει δυναμικά τα δεδομένα στον χρήστη, χωρίς άμεση πρόσβαση στη βάση δεδομένων αλλά μέσω της ελεγχόμενης επίστρωσης του backend.

Για τη μόνιμη αποθήκευση και ανάκτηση δεδομένων, η εφαρμογή Spring Boot αξιοποιεί την PostgreSQL (βλ. Ενότητα 3.6). Μέσω του Spring Data JPA ο ορισμός των οντοτήτων (entity classes) και των αντίστοιχων πινάκων στη βάση γίνεται εύκολα και υποστηρίζονται σύνθετα ερωτήματα (queries) με χρήση της Hibernate Query Language (HQL) ή μέσω Spring Repository μεθόδων. Έτσι, η επιχειρησιακή λογική της εφαρμογής μπορεί να παραμείνει καθαρή και επικεντρωμένη, ενώ οι λεπτομέρειες επικοινωνίας με τη βάση δεδομένων χειρίζονται από το πλαίσιο του Spring.

## Πλεονεκτήματα

Το Spring Boot επιταχύνει την ανάπτυξη του backend χάρη στο auto-configuration και τον ενσωματωμένο HTTP server, ενώ τα annotations διευκολύνουν τον ορισμό καθαρών REST endpoints. Η σύζευξη με Spring Data JPA απλοποιεί την πρόσβαση στην PostgreSQL, και τα εργαλεία logging/Actuator βοηθούν στη λειτουργική παρακολούθηση. Ο σαφής διαχωρισμός controller-service-repository βελτιώνει την επεκτασιμότητα και τη δοκιμασιμότητα και ταιριάζει φυσικά με μονοσέλιδα (Single-Page Application, SPA) frontends. Επιπλέον, μπορεί να επεκταθεί για αμφίδρομη λειτουργία, δηλαδή αν απαιτηθούν downlinks, το backend μπορεί να εκθέτει endpoints που ζητούν από το TTS την αποστολή εντολών (π.χ. ενεργοποίηση actuator).

## Προκλήσεις

Απαιτείται προσεκτικός σχεδιασμός REST και JPA για αποφυγή θεμάτων απόδοσης (N+1, βαριά joins). Η ενημέρωση σε σχεδόν πραγματικό χρόνο μέσω REST συχνά οδηγεί σε polling, συνεπώς πιθανότατα να χρειαστούν SSE/WebSockets. Η κλιμάκωση φέρνει ανάγκες σε ασφάλεια και έλεγχο πρόσβασης, σε σωστή ρύθμιση PostgreSQL (connection pooling,

δείκτες, μεταναστεύσεις σχήματος) και συνεχή παρακολούθηση για σταθερή παραγωγική λειτουργία.

### 3.5 ReactJS

Η React JS είναι μια δημοφιλής βιβλιοθήκη της JavaScript για την ανάπτυξη δυναμικών διεπαφών χρήστη (UI) σε web εφαρμογές, όπου δημιουργήθηκε από τον Jordan Walker, έναν μηχανικό λογισμικού (software engineer) της Facebook (πλέον Meta) και κυκλοφόρησε ως έργο ανοικτού κώδικα το 2013, γνωρίζοντας ευρεία αποδοχή στην κοινότητα των προγραμματιστών. Η React βασίζεται στην αρχιτεκτονική του μονοσέλιδου περιβάλλοντος/Single Page Application (SPA) όπου αντί η εφαρμογή να φορτώνει πολλαπλές ξεχωριστές σελίδες από τον διακομιστή, φορτώνει μία κύρια σελίδα και δυναμικά ενημερώνει το περιεχόμενό της καθώς ο χρήστης αλληλεπιδρά. Αυτό επιτυγχάνεται μέσω του Virtual DOM (Document Object Model), ενός εσωτερικού μηχανισμού της React που διαχειρίζεται αποδοτικά τις αλλαγές στο περιεχόμενο της σελίδας [47].

Στην React, η διεπαφή χρήστη κατασκευάζεται από επαναχρησιμοποιήσιμα συστατικά (components). Κάθε component αντιστοιχεί σε κάποιο τμήμα της οθόνης (π.χ. ένα κουμπί, ένα γράφημα, μια λίστα δεδομένων) και μπορεί να έχει τη δική του κατάσταση (state) και ιδιότητες (props). Όταν αλλάζει η κατάσταση ενός component (για παράδειγμα, όταν φθάνουν νέα δεδομένα αισθητήρων από τον διακομιστή), η React υπολογίζει ποια μέρη του DOM πρέπει να τροποποιηθούν και ενημερώνει μόνο αυτά, αντί να ξαναφορτώσει ολόκληρη τη σελίδα. Αυτό κάνει τις εφαρμογές περισσότερο αποκρίσιμες και βελτιώνει την εμπειρία του χρήστη. Επιπλέον, η React συνδυάζεται συχνά με το Redux ή το Context API για διαχείριση της κατάστασης σε μεγάλη κλίμακα, όταν δηλαδή πολλά components χρειάζεται να μοιράζονται δεδομένα [47].

Στο πλαίσιο του παρόντος συστήματος η React ΘΣ αξιοποιήθηκε για την υλοποίηση του φροντενδ προκειμένου να επιτευχθεί δυναμική και αποδοτική παρουσίαση των δεδομένων. Το περιβάλλον χρήστη που δημιουργήθηκε με τη React επιτρέπει την εποπτεία των δεδομένων που συλλέγονται από τους αισθητήρες και αποστέλλονται μέσω του δικτύου LoRaWAN. Συγκεκριμένα, η React εφαρμογή φορτώνεται στον φυλλομετρητή (browser) του χρήστη και επικοινωνεί με το backend (Spring Boot) μέσω HTTP API calls (με χρήση της fetch API/Axios). Για παράδειγμα, όταν ο χρήστης ανοίγει τη σελίδα, η React καλεί ένα endpoint του backend ώστε να λάβει τις τελευταίες μετρήσεις ή το ιστορικό δεδομένων και αποθηκεύει αυτές τις τιμές στην εσωτερική κατάσταση των components. Στη συνέχεια, τα components αυτά (πίνακας τιμών και γραφήματα) προβάλλουν τα δεδομένα αυτά.

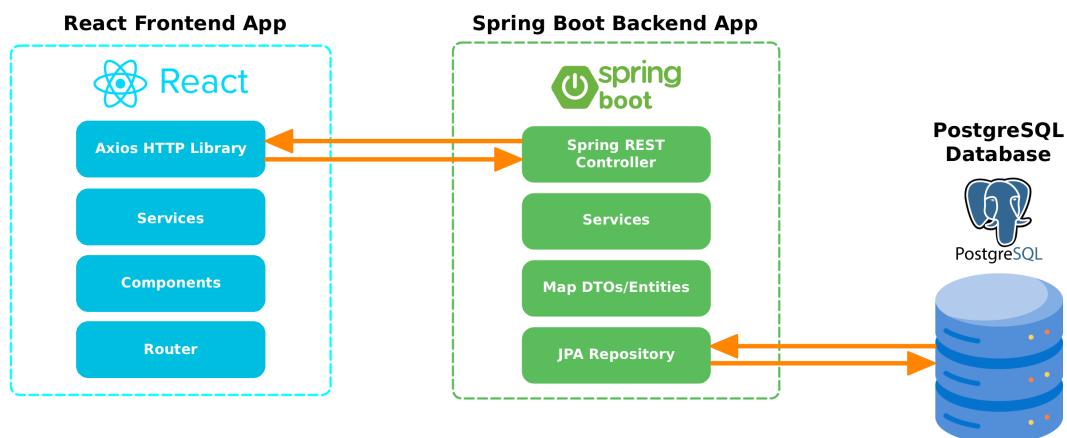
#### Πλεονεκτήματα

Η React διευκολύνει την υλοποίηση πλούσιων διεπαφών με υψηλή απόδοση χάρη στο Virtual DOM και τον αρχιτεκτονικό διαχωρισμό σε επαναχρησιμοποιήσιμα components. Η δομή SPA μειώνει τους χρόνους μετάβασης και προσφέρει ομαλή εμπειρία χρήστης, στοιχείο ιδιαίτερα χρήσιμο για ζωντανή απεικόνιση μετρήσεων. Η επικοινωνία με το backend μέσω REST API ενσωματώνεται απλά με fetch/Axios, ενώ η αξιοποίηση Context API ή βιβλιοθη-

κών κατάστασης (state management) επιτρέπει συνεπή διαχείριση δεδομένων σε όλο το UI. Επιπλέον, με χρήση TypeScript (γλώσσα προγραμματισμού με αυστηρό τύπο που βασίζεται και επεκτείνει τη JavaScript) η επιβολή τύπων βελτιώνει την αξιοπιστία και τη συντηρησιμότητα του κώδικα. Το πλούσιο οικοσύστημα (charting, UI kits) επιταχύνει την προσθήκη γραφημάτων και πινάκων για την οπτικοποίηση των αισθητήρων.

## Προκλήσεις

Η κλιμακούμενη διαχείριση κατάστασης μπορεί να γίνει σύνθετη, ειδικά όταν πολλά components μοιράζονται δεδομένα ή απαιτείται ενημέρωση σχεδόν πραγματικού χρόνου, όπου ίσως χρειαστούν SSE/WebSockets αντί για απλό polling (περιοδικά αιτήματα για δεδομένα από το backend). Η απόδοση θέλει προσοχή σε μεγάλες λίστες ή συχνές ενημερώσεις (memoization, virtualization). Επιπλέον, θέματα όπως Search Engine Optimization (SEO) σε SPAs χωρίς Server-Side Rendering (SSR) και ο σωστός χειρισμός σφαλμάτων/φορτώσεων (error and loading states) απαιτούν επιπρόσθετο σχεδιασμό. Τέλος, το σύγχρονο tooling (bundlers και ρυθμίσεις παραγωγής) εισάγει πολυπλοκότητα που πρέπει να ρυθμιστεί εξ αρχής σωστά για βέλτιστη απόδοση και ασφάλεια.



Εικόνα 3.4: Αρχιτεκτονική ενός Web Application, αποτελούμενο από ένα React Frontend App που επικοινωνεί μέσω Axios/REST APIs με ένα Spring Boot Backend App, μετατρέπει δεδομένα DTOs (Data Transfer Objects) σε οντότητες (Entities) και εκτελεί διεργασίες απόθηκευσης/ανάκτησης αυτών των δεδομένων από μία PostgreSQL βάση δεδομένων.

## 3.6 PostgreSQL

To PostgreSQL είναι ένα ισχυρό σύστημα διαχείρισης σχεσιακών δεδομένων (Relational Database Management System, RDBMS) ανοικτού κώδικα, το οποίο χρησιμοποιείται ευρέως σε εφαρμογές που απαιτούν αξιοπιστία, συνέπεια και απόδοση στην αποθήκευση

δεδομένων. Προέκυψε ως απόγονος του έργου POSTGRES στο Πανεπιστήμιο της Καλιφόρνια το 1986 και έκτοτε έχει εξελιχθεί σε ένα από τα πιο προηγμένα διαθέσιμα RDBMS, υποστηρίζοντας μεγάλο μέρος του προτύπου SQL:2011 και παρέχοντας πλήθος επεκτάσεων. Το PostgreSQL γνωστό για τη συμμόρφωσή του με τις αρχές ACID (Atomicity, Consistency, Isolation, Durability), εγγυάται την αξιόπιστη εκτέλεση των συναλλαγών σε μια PostgreSQL βάση δεδομένων και τη διατήρηση της ακεραιότητάς τους ακόμη και σε περίπτωση σφαλμάτων ή ταυτόχρονων προσπελάσεων [48].

Ένα από τα χαρακτηριστικά που ξεχωρίζουν το PostgreSQL είναι η δυνατότητα επέκτασής του. Υποστηρίζει προσαρμοσμένους τύπους δεδομένων, συναρτήσεις, ακόμα και αποθηκευμένες διαδικασίες σε διάφορες γλώσσες προγραμματισμού. Επίσης, διαθέτει υποστήριξη για αποθήκευση JSON δεδομένων και εκτέλεση ερωτημάτων (queries) πάνω σε αυτά, γεγονός που το καθιστά ικανό να λειτουργεί κατά περίπτωση και ως NoSQL. Η μηχανή ευρετηρίων του PostgreSQL είναι ιδιαίτερα προηγμένη, προσφέροντας πολλούς τύπους ευρετηρίων (B-tree, Hash, GIN, GiST, BRIN) που μπορούν να βελτιστοποιήσουν την απόδοση σε διαφορετικά είδη ερωτημάτων [48].

Στο σύστημα που αναπτύξαμε, το PostgreSQL αποτέλεσε το κύριο αποθετήριο δεδομένων. Συγκεκριμένα, εγκαταστάθηκε μια PostgreSQL βάση δεδομένων για την αποθήκευση τόσο των μεταδεδομένων του δικτύου LoRaWAN όσο και των δεδομένων της εφαρμογής. Από την πλευρά του The Things Stack, η βάση αυτή χρησιμοποιείται για την τήρηση των απαραίτητων πληροφοριών: εγγραφές συσκευών (π.χ. DevEUI, AppKey, κ.λπ.), στοιχεία gateway (π.χ. EUI και κλειδιά), λογαριασμοί χρηστών και δικαιώματα, καθώς και τα session keys και άλλες λεπτομέρειες που απαιτούνται για τη λειτουργία του LoRaWAN Network Server. Το TTS έχει σχεδιαστεί ώστε να είναι αινεξάρτητο από το είδος της βάσης (υποστηρίζει και άλλες SQL βάσεις ή embedded SQLite), όμως η χρήση της PostgreSQL βάσης συνιστάται σε παραγωγικό περιβάλλον λόγω της σταθερότητας και των δυνατοτήτων κλιμάκωσης που προσφέρει.

Παράλληλα, από την πλευρά της εφαρμογής Spring Boot, το PostgreSQL χρησιμοποιείται για την αποθήκευση των δεδομένων των αισθητήρων και λοιπών πληροφοριών της εφαρμογής. Έχει δημιουργηθεί το αντίστοιχο σχήμα (schema) με πίνακες που περιλαμβάνουν τους μετρητές (meters), τις μετρήσεις ανά αισθητήρα/φάση (sensor\_data) και χρήστες για αυθεντικοποίηση (users). Το Spring Boot συνδέεται στη βάση χρησιμοποιώντας οδηγό JDBC (Java Database Connectivity) και μέσω του JPA εκτελεί τις κατάλληλες συναλλαγές για εισαγωγή νέων μετρήσεων ή ανάκτηση ιστορικών δεδομένων. Για παράδειγμα, όταν φτάνει ένα νέο webhook μετρήσεων, δημιουργείται μια νέα εγγραφή στον πίνακα μετρήσεων με τα αντίστοιχα πεδία, ενώ όταν το frontend ζητά το ιστορικό, εκτελείται ένα SELECT ερώτημα στη βάση που επιστρέφει τις εγγραφές για τον συγκεκριμένο αισθητήρα.

## Πλεονεκτήματα

Η PostgreSQL θεωρείται από τα πιο αξιόπιστα και σταθερά RDBMS (Relational Database Management System), με ενεργή κοινότητα και μακρά ιστορία ανάπτυξης. Υποστηρίζει πλήρως τις αρχές ACID (Atomicity, Consistency, Isolation, and Durability), εξασφαλίζοντας ακεραιότητα και συνέπεια στα δεδομένα, γεγονός κρίσιμο για εφαρμογές όπου η ακρίβεια των μετρήσεων είναι ουσιαστική. Η δυνατότητα επεκτασιμότητας (π.χ. προσαρμοσμένοι τύποι

δεδομένων, extensions όπως το PostGIS) την καθιστά ιδιαίτερα ευέλικτη. Επιπλέον, η ύπαρξη πολλών μηχανισμών ευρετηρίων συμβάλλει σε βελτιστοποιημένη απόδοση ακόμη και σε μεγάλους όγκους δεδομένων. Στο πλαίσιο της εφαρμογής μας, τα πλεονεκτήματα της PostgreSQL ενισχύουν την αξιοπιστία του συστήματος, καθώς τα δεδομένα των μετρητών και των αισθητήρων αποθηκεύονται με ασφάλεια και μπορούν να ανακτηθούν αποτελεσματικά μέσω του Spring Data JPA. Η αρχιτεκτονική της βάσης επιτρέπει επίσης την εύκολη υποστήριξη επιπλέον οντοτήτων (π.χ. νέοι τύποι αισθητήρων) χωρίς σημαντικές αλλαγές στον κώδικα. Τέλος, η χρήση της ίδιας βάσης τόσο από το The Things Stack όσο και από το backend εξασφαλίζει ομοιομορφία και αποφεύγει την ανάγκη πολλαπλών επιπέδων αποθήκευσης.

## Προκλήσεις

Παρότι η PostgreSQL είναι ιδιαίτερα ισχυρή, η σωστή παραμετροποίησή της μπορεί να απαιτεί εμπειρία, ειδικά σε περιπτώσεις μεγάλης κλίμακας με υψηλά φορτία. Για παράδειγμα, η διαχείριση ταυτόχρονων συνδέσεων ή η ρύθμιση της μνήμης (work\_mem, shared\_buffers) επηρεάζουν σημαντικά την απόδοση. Στο πλαίσιο της εφαρμογής μας, καθώς οι μετρήσεις αυξάνονται, είναι πιθανό να χρειαστούν βελτιστοποιήσεις σε επίπεδο ευρετηρίων και ερωτημάτων για την αποφυγή καθυστερήσεων. Επιπλέον, η συνδυαστική χρήση της βάσης από το TTS και το Spring Boot backend απαιτεί προσοχή στη σχεδίαση του schema, ώστε να αποφευχθούν συγκρούσεις ή περιττή πολυπλοκότητα. Τέλος, όπως σε κάθε on-premises βάση, η συντήρηση (ενημερώσεις, αντίγραφα ασφαλείας, παρακολούθηση απόδοσης) αποτελεί αναγκαία διαδικασία για τη διασφάλιση της απρόσκοπτης λειτουργίας του συστήματος.

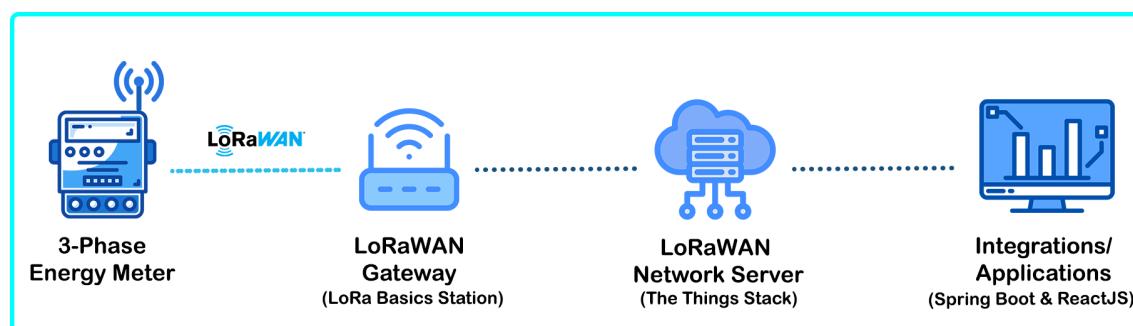
## Κεφάλαιο 4

# Υλικό και σχεδιασμός συσκευών του συστήματος

Σε αυτό το κεφάλαιο περιγράφεται η υλική υποδομή που κατασκευάστηκε για την υλοποίηση του συστήματος, το **LoRaWAN gateway** και οι δύο **τριφασικοί μετρητές**. Δίνεται έμφαση στον σχεδιασμό, στα υλικά και στα χαρακτηριστικά τους, καθώς και στη διασύνδεση και στις τεχνικές επιλογές.

### 4.1 Γενική αρχιτεκτονική συστήματος

Πριν από την περιγραφή κάθε συσκευής, αξίζει να παρουσιαστεί συνοπτικά η τοπολογία: οι αισθητήρες (μετρητές) επικοινωνούν ασύρματα μέσω LoRaWAN με το gateway, το οποίο συλλέγει τα πακέτα και τα προωθεί (forwarding) προς τον LoRaWAN Network Server. Από εκεί, τα δεδομένα διατρέχουν το backend μέχρι την τελική αποθήκευση και παρουσίαση στο frontend της εφαρμογής μας.



Εικόνα 4.1: Τοπολογία του συστήματος.

### 4.2 LoRaWAN Gateway

#### 4.2.1 Υλικά και κατασκευή

Για την κατασκευή του gateway χρησιμοποιήθηκε ένα **Raspberry Pi 4 Model B** σε συνδυασμό με τη μονάδα συγκεντρωτή (concentrator board) **iC880A-SPI** της IMST. Σύμφωνα με τον οδηγό της The Things Industries [49], αυτές οι συνθέσεις υποστηρίζονται ώστε να

κατασκευάζει κανείς ένα προσωπικό gateway χρησιμοποιώντας τα προαναφερθέντα εξαρτήματα.

Ο ακόλουθος πίνακας συνοψίζει τα βασικά υλικά που χρησιμοποιήθηκαν:

Συσκευή / Στοιχείο	Σκοπός / Σχόλιο
1× Raspberry Pi 4 Model B	Κεντρικός υπολογιστής και host λογισμικού
1× Concentrator board iC880A-SPI	Διαχείριση πακέτων LoRa σε πολλαπλά κανάλια
1× Κεραία 868MHz 2dBi	Εκπομπή / λήψη ραδιοσημάτων
1× Pigtail καλώδιο	Σύνδεση κεραίας με iC880A-SPI
7× Jumper wires, dual female	Σύνδεση SPI, reset, power lines
1× Κάρτα microSD, 64GB	Λειτουργικό σύστημα και λογισμικό
1× Τροφοδοσία (5V, $\geq 3A$ )	Παροχή, αναγκαία για σταθερή λειτουργία

Πίνακας 4.1: Βασικά υλικά για το LoRaWAN Gateway.

## Raspberry Pi 4 Model B

Το Raspberry Pi 4 Model B (βλ. Εικόνα 4.2) αποτελεί μια ευέλικτη και ευρύτατα διαδεδομένη πλατφόρμα μικροϋπολογιστή (single-board computer) που χρησιμοποιείται ευρέως σε έργα IoT, αυτοματισμού και πρωτότυπης ανάπτυξης. Στηρίζεται σε ανοικτό λογισμικό και κοινοτικά πρότυπα, επιτρέποντας τη χρήση του σε πειραματικά ή παραγωγικά έργα με ευελιξία και χαμηλό κόστος. Υποστηρίζει πλήρες Linux οικοσύστημα και συνοδεύεται από μια πολύ ενεργή κοινότητα, γεγονός που το καθιστά ιδιαίτερα ελκυστικό για μαθητές και επαγγελματίες που θέλουν να υλοποιήσουν έργα IoT, αυτοματισμού ή γρήγορου prototyping [50]. Στο πλαίσιο της παρούσας υλοποίησης, το Raspberry Pi 4 χρησιμεύει ως κόμβος που «υποδέχεται» και διαχειρίζεται το λογισμικό του gateway, συμβάλλοντας καθοριστικά στην αξιοποιησία και επεκτασιμότητα του συστήματος. Επιπλέον, αποτελεί και το host μηχάνημα όπου θα φιλοξενηθούν τόσο το λογισμικό του LoRaWAN network server όσο και η τελική εφαρμογή του χρήστη.

## LoRaWAN Concentrator board iC880A-SPI

Το iC880A-SPI (βλ. Εικόνα 4.2) είναι συγκεντρωτής LoRaWAN που χρησιμοποιείται ως το κύριο στοιχείο ενός gateway, δηλαδή ως πλήρες RF frontend που δέχεται ταυτόχρονα πολλαπλά LoRa πακέτα σε διαφορετικά κανάλια και προωθεί την κίνηση προς τον network server. Τοποθετείται συνήθως πάνω σε single-board πλακέτες όπως το Raspberry Pi και έχει καθιερωθεί σε DIY και εργαστηριακές υλοποιήσεις λόγω της ευρείας τεκμηρίωσης και της συμβατότητάς του με σύγχρονα λογισμικά gateway όπως το LoRa Basics Station. Με αυτό τον τρόπο επιτρέπει την κατασκευή οικονομικών αλλά αξιόπιστων LoRaWAN gateways που εντάσσονται εύκολα σε δίκτυα όπως το The Things Stack [51].

Τεχνικά Δεδομένα	Περιγραφή
Επεξεργαστής	Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Μνήμη	2GB
Συνδεσιμότητα	2.4GHz and 5.0GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports, 2× USB 2.0 ports
Είσοδοι/ Έξοδοι	Standard 40-pin GPIO header
Ήχος και Εικόνα	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Πολυμέσα	H.265 (4Kp60 decode) H.264 (1080p60 decode, 1080p30 encode) OpenGL ES, 3.0 graphics
Υποστήριξη κάρτας SD	Micro SD card slot for loading operating system and data storage
Ισχύς Εισόδου	5V DC via USB-C connector (minimum 3A) 5V DC via GPIO header (minimum 3A) Power over Ethernet (PoE)-enabled (requires separate PoE HAT)
Περιβάλλον λειτουργίας	Operating temperature 0-50°C

Πίνακας 4.2: Τεχνικά χαρακτηριστικά Raspberry Pi 4 Model B.

[50]

Τεχνικά Δεδομένα	Περιγραφή
Βασικό Chipset	Semtech SX1301 baseband processor + 2× SX1257 RF transceivers
Συχνότητες	EU868 (διαθέσιμη έκδοση και για US915)
Κανάλια	8× LoRa channels ταυτόχρονα + 1× FSK channel
Διεπαφή προς Host	SPI (3.3V logic level)
Χρονισμός	Είσοδος PPS από GPS για συγχρονισμό
Σύνδεση Κεραίας	Υποδοχή SMA 50Ω
Τάση / Ισχύς	5V μέσω ακροδεκτών (κατανάλωση ~ 700mW)
Εφαρμογή	Συγκεντρωτής (concentrator) για LoRaWAN gateways
Διαστάσεις	80mm × 50mm περίπου

Πίνακας 4.3: Τεχνικά χαρακτηριστικά του iC880A-SPI LoRaWAN Concentrator.

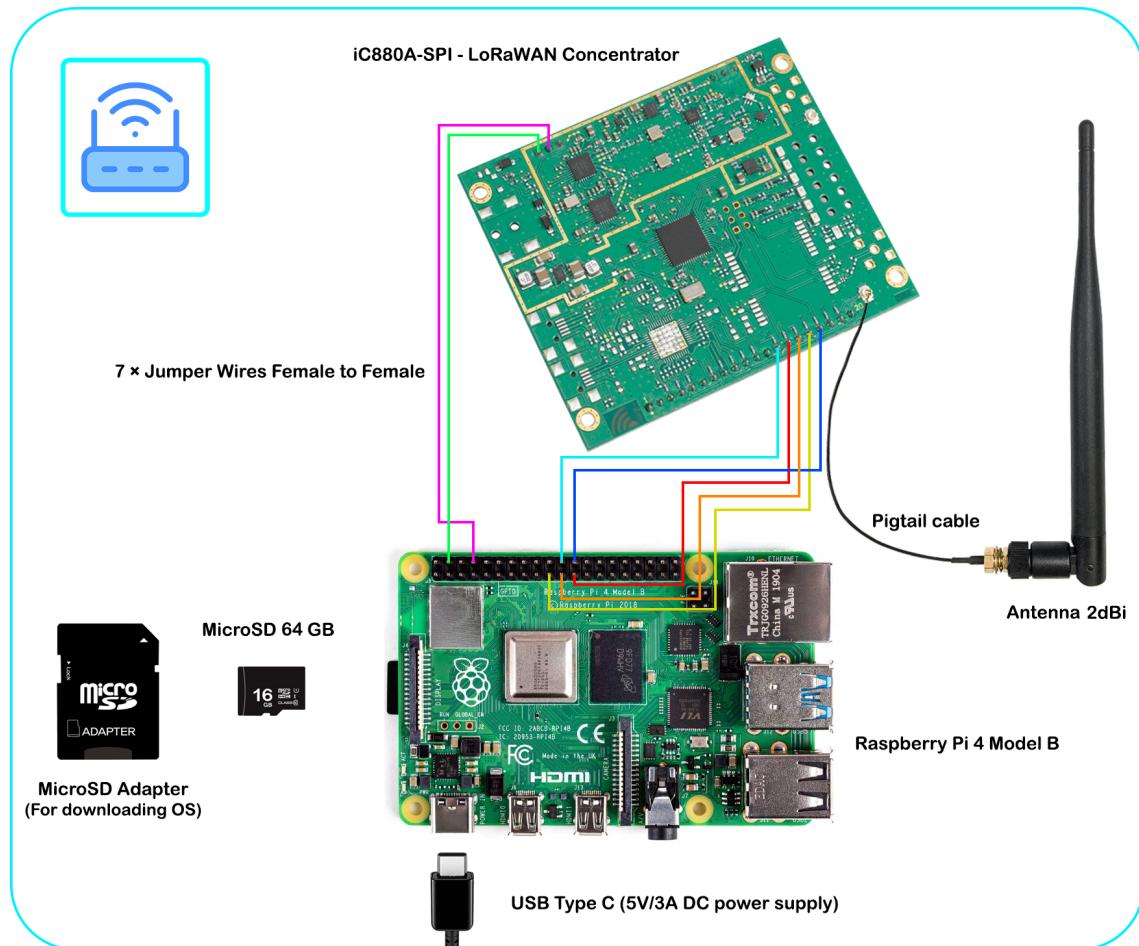
[51]

### 4.2.2 Συνδεσμολογία και λειτουργία

Η συνδεσμολογία μεταξύ Raspberry Pi και συγκεντρωτή IC880A-SPI ακολουθεί τις οδηγίες που παρέχει η The Things Industries: το board συνδέεται μέσω SPI pins (MOSI, MISO, SCLK, NSS) καθώς και reset, 5V και γείωσης (GND). Η συνδεσμολογία φαίνεται στην Εικόνα 4.2 και περιγράφεται σε επίπεδο ακίδων (pins) στον Πίνακα 4.4.

Πριν την ενεργοποίηση, είναι απαραίτητο να συνδεθεί η κεραία στο concentrator, προκειμένου να αποφευχθεί η ανάκλαση ισχύος που θα μπορούσε να βλάψει το hardware.

Το λογισμικό του gateway (LoRa Basics Station) εκτελείται στο Raspberry Pi και επικοινωνεί με τον συγκεντρωτή, στέλνοντας uplink μηνύματα προς τον TTS και λαμβάνοντας τυχόν downlink. Στη συνέχεια τα δεδομένα αυτά στέλνονται και παρουσιάζονται στην εφαρμογή μας. Οι υπηρεσίες αυτές θα μπορούσαν κάλλιστα να φιλοξενούνται σε διαφορετικές συσκευές (servers), εντούτοις στη δική μας υλοποίηση επιλέχθηκε να φιλοξενηθούν με χρήση του Docker (βλ. Ενότητα 3.3) όλες στο ίδιο μηχάνημα (LoRaWAN gateway) για λόγους απλοποίησης και εξοικονόμησης υλικού.



Εικόνα 4.2: Υλοποίηση LoRaWAN Gateway και σύνδεσμολογία εξαρτημάτων.

iC880A pin	Raspberry Pi pin	Description
21	2	5V power supply
22	6	GND
13	22	Reset
14	23	SPI Clock
15	21	MISO
16	19	MOSI
17	24	NSS

Πίνακας 4.4: Συνδεσομολογία ακίδων (pins) iC880A-SPI με Raspberry Pi 4 Model B.

## 4.3 Τριφασικοί Μετρητές

### 4.3.1 Υλικά και αρχιτεκτονική

Οι τριφασικοί μετρητές που κατασκευάστηκαν έχουν ως στόχο την καταγραφή ηλεκτρικών μεγεθών ανά φάση και τη μετάδοση των μετρήσεων μέσω LoRaWAN προς το gateway. Κάθε μετρητής αποτελείται από μία πλακέτα **The Things Uno** ως κεντρικό μικροελεγκτή και LoRaWAN πομπό, καθώς και από τρεις αισθητήρες **PZEM-004T V3.0** (έναν για κάθε φάση). Οι PZEM-004T μετρούν τάση, ρεύμα, ενεργό ισχύ, ενέργεια, συντελεστή ισχύος και συχνότητα, εκθέτοντας τις τιμές μέσω σειριακής TTL/Modbus-RTU. Οι μετρήσεις συλλέγονται από τη The Things Uno και αποστέλλονται ασύρματα μέσω LoRaWAN στο δίκτυο.

Ο ακόλουθος πίνακας συνοψίζει τα βασικά υλικά που χρησιμοποιήθηκαν για την κατασκευή ενός τριφασικού μετρητή. Για την παρούσα υλοποίηση κατασκευάστηκαν δύο όμοιες μονάδες.

Συσκευή / Στοιχείο	Σκοπός / Σχόλιο
1 × The Things Uno	Κεντρικός μικροελεγκτής ATmega32U4 με LoRaWAN μονάδα Microchip RN2483/RN2903.
3 × PZEM-004T V3.0	Μετρητική μονάδα AC ανά φάση με έξοδο TTL/Modbus-RTU.
3 × Current Transformers (CT)	Διακύλιοι ρεύματος κατάλληλοι για τη σειρά PZEM-004T V3.0 (π.χ. 100A).
3 × Καλωδιώσεις TTL και ισχύος	Σύνδεση TX/RX/GND/5V προς τους PZEM και τροφοδοσία πλακετών.

Πίνακας 4.5: Κατάλογος υλικών για έναν τριφασικό μετρητή

### PZEM-004T V3.0

Ο PZEM-004T V3.0 (βλ. Εικόνα 4.3) είναι ένας πολυλειτουργικός ψηφιακός μετρητής εναλλασσόμενης ενέργειας (Alternating Current, AC), σχεδιασμένος για τη μέτρηση τάσης,

ρεύματος, ενεργού ισχύος, συχνότητας, συντελεστή ισχύος και της συσσωρευμένης κατανάλωσης ενέργειας σε μονοφασικά συστήματα εναλλασσόμενου ρεύματος. Εκθέτει τις τιμές μέσω TTL με Modbus-RTU σε 9600bps εξ ορισμού. Υπάρχουν παραλλαγές για 10A (με εσωτερικό shunt) και 100A (με εξωτερικό CT) [52].

Παράμετρος	Ενδεικτικές προδιαγραφές V3.0
Εύρος τάσης	80-260VAC
Εύρος ρεύματος	0-100A με εξωτερικό CT
Μετρούμενες ποσότητες	Τάση, ρεύμα, ενεργός ισχύς, ενέργεια, συχνότητα, power factor
Διεπαφή επικοινωνίας	TTL/Modbus-RTU, 9600bps προεπιλογή
Τροφοδοσία	Από την πλευρά της τάσης AC εισόδου
Τυπική ακρίβεια	±0.5-1% για βασικές μετρήσεις (ενδεικτικό, ανά τεκμηρίωση V3.0)

Πίνακας 4.6: Τεχνικά χαρακτηριστικά PZEM-004T V3.0  
[52]

### The Things Uno

H The Things Uno (βλ. Εικόνα 4.3) της εταιρείας The Things Industries βασίζεται στο Arduino Leonardo με μικροελεγκτή ATmega32U4 και ενσωματωμένη μονάδα LoRaWAN Microchip RN2483 (για EU868) ή RN2903 (για US915). Είναι πλήρως συμβατή με το Arduino IDE και τις υπάρχουσες Arduino shields, ενώ στο επίπεδο λογισμικού η επικοινωνία με τη μονάδα LoRa γίνεται μέσω της Serial1. H πλατφόρμα προσφέρεται για γρήγορο prototyping και εύκολη ένταξη με το λογισμικό The Things Stack [53].

Στοιχείο	Περιγραφή
Κεντρικός MCU	ATmega32U4 (Arduino Leonardo-compatible)
LoRaWAN μονάδα	Microchip RN2483 (EU868) ή RN2903 (US915)
Προγραμματισμός	Arduino IDE, TheThingsNetwork βιβλιοθήκη, Serial1 προς RN2483/RN2903
Διεπαφές	Ψηφιακές I/O, UART, SPI, I2C, USB
Τροφοδοσία	7-9V DC power adapter με 2.1mm jack plug ή 5V micro USB
Ενδεικτική εμβέλεια	Μέχρι και 10 χιλιόμετρα, ανάλογα με περιβάλλον και κεραία

Πίνακας 4.7: Τεχνικά χαρακτηριστικά του The Things Uno

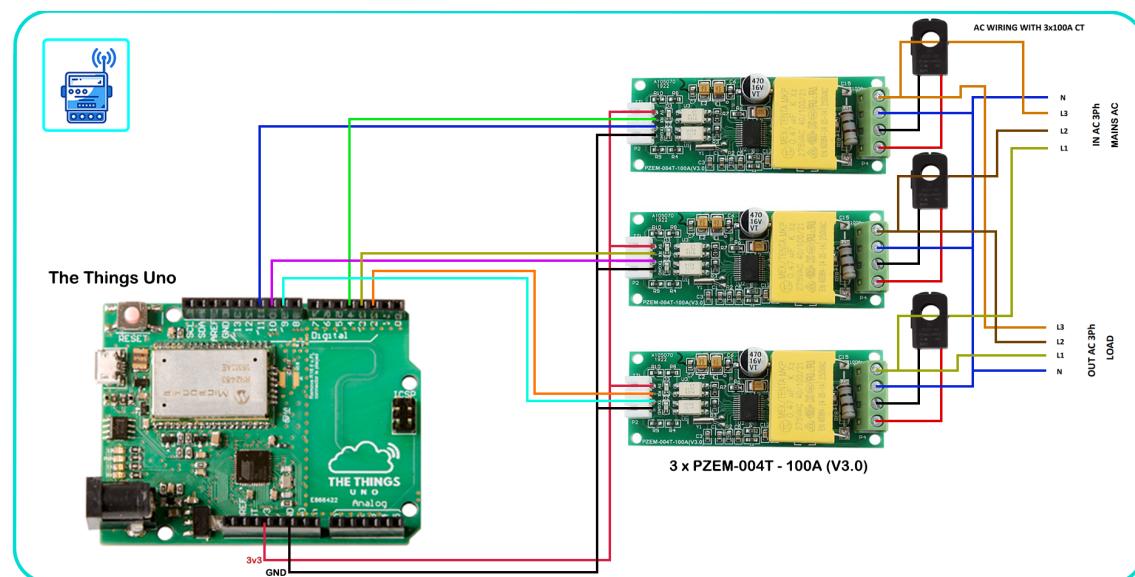
#### 4.3.2 Συνδεσμολογία και λειτουργία

H The Things Uno συλλέγει τις τιμές από τις τρεις μονάδες PZEM, δημιουργεί ένα συνολικό πακέτο δεδομένων (τρεις φάσεις) και στέλνει ασύρματα μέσω LoRaWAN στο gateway.

Κατά την ένταξη, η συσκευή πραγματοποιεί join procedure με τον Join Server του TTS και λαμβάνει τα κλειδιά συνεδρίας για ασφαλή διαδρομή δεδομένων. Οι μετρήσεις αποστέλλονται περιοδικά με προκαθορισμένο διάστημα.

**Σημείωση:** Επιλέχθηκε η τροφοδοσία των PZEM-004T v3.0 από το 3v3 (3.3V) της The Things Uno αντί για 5V, διότι στην πράξη η λειτουργία στα 5V παρήγαγε μη έγκυρες μετρήσεις/σφάλματα στη UART. Η The Things Uno λειτουργεί σε λογικά επίπεδα 3.3V, επομένως όταν τα PZEM τροφοδοτούνται στα 5V εκπέμπουν TX σήμα κοντά στα 5V, κάτι που μπορεί να υπερβεί τα επιτρεπτά όρια του RX της πλακέτας και να οδηγήσει σε framing errors. Με τροφοδοσία στα 3.3V τα επίπεδα ευθυγραμμίζονται με ασφάλεια με τη UART του μικροελεγκτή και οι μετρήσεις σταθεροποιούνται, χωρίς ανάγκη για ξεχωριστούς level shifters.

Η εικόνα παρακάτω δείχνει τη συνδεσμολογία μεταξύ της πλακέτας The Things Uno και των τριών PZEM modules:



Εικόνα 4.3: Υλοποίηση τριφασικού μετρητή - σύνδεση The Things Uno με 3 PZEM-004T V3.0

PZEM 1 (Phase L1)		
PZEM-004T v3.0 pin	The Things Uno pin	Description
VCC	3v3V	Τροφοδοσία 3.3V
GND	GND	Κοινή γείωση (shared ground)
TX	D9	PZEM TX → Uno RX (SoftwareSerial RX)
RX	D2	PZEM RX → Uno TX (SoftwareSerial TX)

Πίνακας 4.8: Συνδεσμολογία 1<sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 1

<b>PZEM 2 (Phase L2)</b>		
<b>PZEM-004T v3.0 pin</b>	<b>The Things Uno pin</b>	<b>Description</b>
VCC	3v3V	Τροφοδοσία 3.3V
GND	GND	Κοινή γείωση (shared ground)
TX	D10	PZEM TX → TTU RX (SoftwareSerial RX)
RX	D3	PZEM RX → TTU TX (SoftwareSerial TX)

Πίνακας 4.9: Συνδεσμολογία 2<sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 2

<b>PZEM 3 (Phase L3)</b>		
<b>PZEM-004T v3.0 pin</b>	<b>The Things Uno pin</b>	<b>Description</b>
VCC	3v3V	Τροφοδοσία 3.3V
GND	GND	Κοινή γείωση (shared ground)
TX	D11	PZEM TX → TTU RX (SoftwareSerial RX)
RX	D4	PZEM RX → TTU TX (SoftwareSerial TX)

Πίνακας 4.10: Συνδεσμολογία 3<sup>ου</sup> PZEM-004T v3.0 με The Things Uno - Φάση 3

**Σημείωση για την επιλογή θυρών (RX):** Στο The Things Uno (Leonardo/ATmega32U4) η βιβλιοθήκη SoftwareSerial (βλ. Υπο-υποενότητα 5.2.2.2) μπορεί να λάβει (RX) μόνο από συγκεκριμένα ψηφιακά pins που υποστηρίζουν pin-change interrupts. Σύμφωνα με την τεκμηρίωση για Leonardo/Micro, τα έγκυρα RX pins είναι τα **8, 9, 10, 11, 14(MISO), 15(SCK), 16(MOSI)**. Γι' αυτό επιλέξαμε τις **D9, D10, D11** για τη σύνδεση PZEM TX - Uno RX, ενώ τα TX της SoftwareSerial τοποθετήθηκαν στις **D2, D3, D4**. Έτσι αποφεύγουμε διενέξεις με το υλικό UART (**Serial1** στους **D0/D1**), το οποίο κρατάμε διαθέσιμο για τη LoRa/TTS επικοινωνία και ταυτόχρονα εξασφαλίζουμε αξιόπιστο χρονισμό σε τυπικές baud rates (π.χ. 9600-38400). Βλ. και τις οδηγίες Arduino για SoftwareSerial και The Things Uno (προφίλ Leonardo) [39, 54, 55].

Στην πλευρά των βιδωτών ακροδεκτών (screw terminals, δεξία μερία των PZEM στην Εικόνα 4.3) το PZEM-004T v3.0 υπάρχουν δύο ζεύγη επαφών: το ζεύγος φάσης/ουδέτερου (**Live/Neutral**) για την τάση από το δίκτυο και το ζεύγος εισόδου **CT (+/-)** για τον μετασχηματιστή ρεύματος. Η φάση L και ο ουδέτερος N συνδέονται απευθείας στα αντίστοιχα καλώδια του τριφασικού/μονοφασικού κλάδου που μετράμε, ώστε το module να μετράει την τάση (voltage) και να τροφοδοτείται. Ο CT «αγκαλιάζει» μόνο τον αγωγό της φάσης (όχι τον ουδέτερο) και τα δύο του καλώδια πηγαίνουν στους ακροδέκτες CT (+) και CT (-) του PZEM ώστε το module να μετράει το ρεύμα (current). Φροντίζουμε ο προσανατολισμός να ταιριάζει με το βέλος/σήμανση του CT ώστε η φορά ισχύος να είναι σωστή. Το PZEM υπολογίζει στη συνέχεια τις μετρήσεις power, energy, frequency και power factor.

**Σημείωση:** Λόγω της αρχιτεκτονικής του PZEM-004T v3.0, οι μετρήσεις ρεύματος/ισχύος δεν φέρουν πρόσημο ούτε πληροφορία γωνίας φάσης, επομένως **δεν είναι δυνατή η διάκριση κατεύθυνσης ροής** (εισροή/εκροή) και η «αντίστροφη» ισχύς δεν μπορεί να αναγνωριστεί από τη συσκευή. Τέλος, το PZEM-004T v3.0 **δεν υπολογίζει αρμονικές ούτε THD**, αλλά εκθέτει μόνο RMS τιμές τάσης/ρεύματος, ενεργό ισχύ, ενέργεια, συχνότητα και power factor, χωρίς φασματική ανάλυση.



## Μέρος II

### Πρακτικό Μέρος



## Κεφάλαιο 5

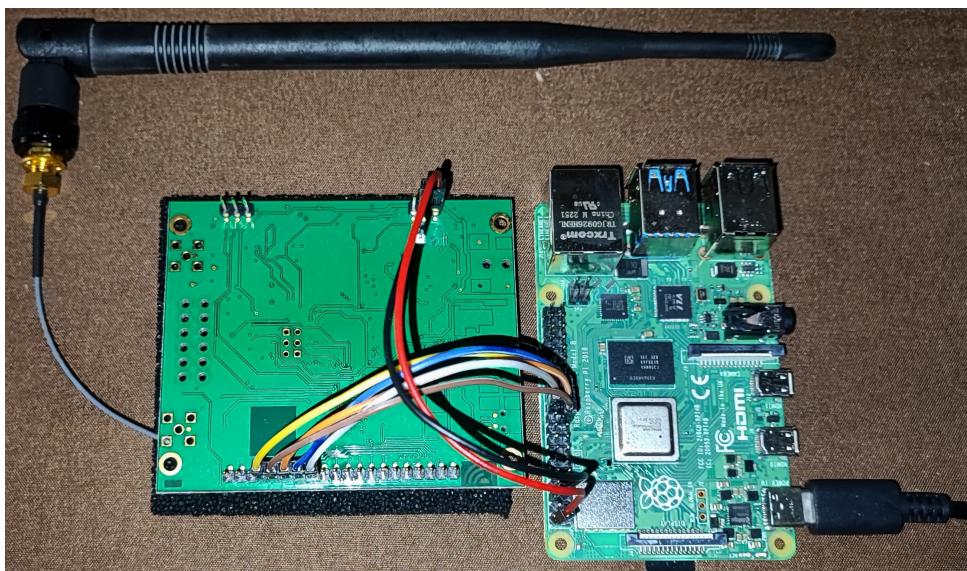
# Πρακτική υλοποίηση του συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται βήμα προς βήμα η πρακτική υλοποίηση του συστήματος, από την αρχική εγκατάσταση μέχρι την πλήρη ολοκλήρωση (integration) των υποσυστημάτων. Αφετηρία αποτελεί η προετοιμασία του LoRaWAN gateway, το οποίο φιλοξενεί με χρήση Docker τις υπηρεσίες The Things Stack και LoRa Basics Station, με αναλυτικές ρυθμίσεις, εντολές και επιλογές παραμετροποίησης ώστε να συνδεθεί με ασφάλεια και να λειτουργεί αξιόπιστα. Στη συνέχεια τεκμηριώνεται ο προγραμματισμός των τριφασικών μετρητών και η σύνδεσή τους με το gateway, με έμφαση στη ροή των δεδομένων και τον τρόπο διαμόρφωσης των uplinks. Τέλος, περιγράφεται η υλοποίηση της τελικής web εφαρμογής και η ανάπτυξή της στο ίδιο gateway, ολοκληρώνοντας μια ενιαία υποδομή από το πεδίο μέχρι το περιβάλλον χρήστη.

## 5.1 Προετοιμασία LoRaWAN gateway

### 5.1.1 Διασύνδεση Hardware

Αρχικά ξεκινάμε με την σύνδεση των επιμέρους εξαρτημάτων όπως περιγράφηκε στην Ενότητα 4.2, σύμφωνα με την Εικόνα 4.2 και τον Πίνακα 4.4:

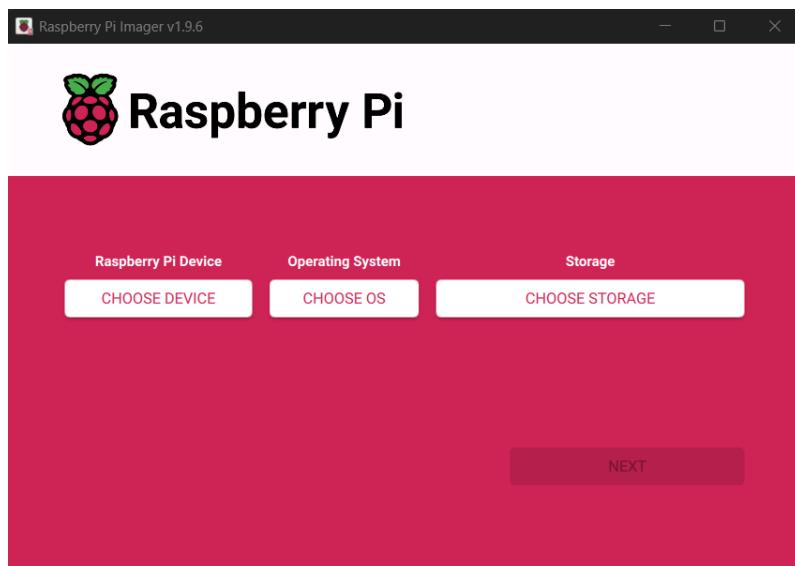


Εικόνα 5.1: Τελική συνδεσμολογία των εξαρτημάτων του LoRaWAN gateway.

### 5.1.2 Εγκατάσταση λειτουργικού συστήματος

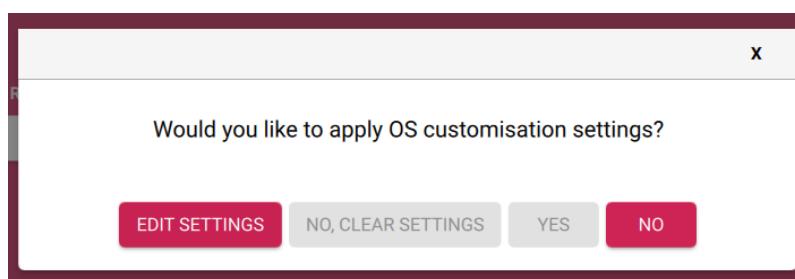
Για την αρχική προετοιμασία του Raspberry Pi απαιτείται η εγκατάσταση του λειτουργικού συστήματος στην κάρτα microSD. Η διαδικασία μπορεί να γίνει πλήρως headless χωρίς οθόνη ή πληκτρολόγιο, αξιοποιώντας το εργαλείο Raspberry Pi Imager. Ακολουθούν τα απαραίτητα βήματα, με πρόσθετες ρυθμίσεις ώστε το σύστημα να ξεκινήσει έτοιμο για δικτυακή πρόσβαση και παραμετροποίηση.

1. Συνδέουμε την κάρτα microSD στον υπολογιστή μέσω card reader και ανοίγουμε το Raspberry Pi Imager.



Εικόνα 5.2: *Raspberry Pi Imager*.

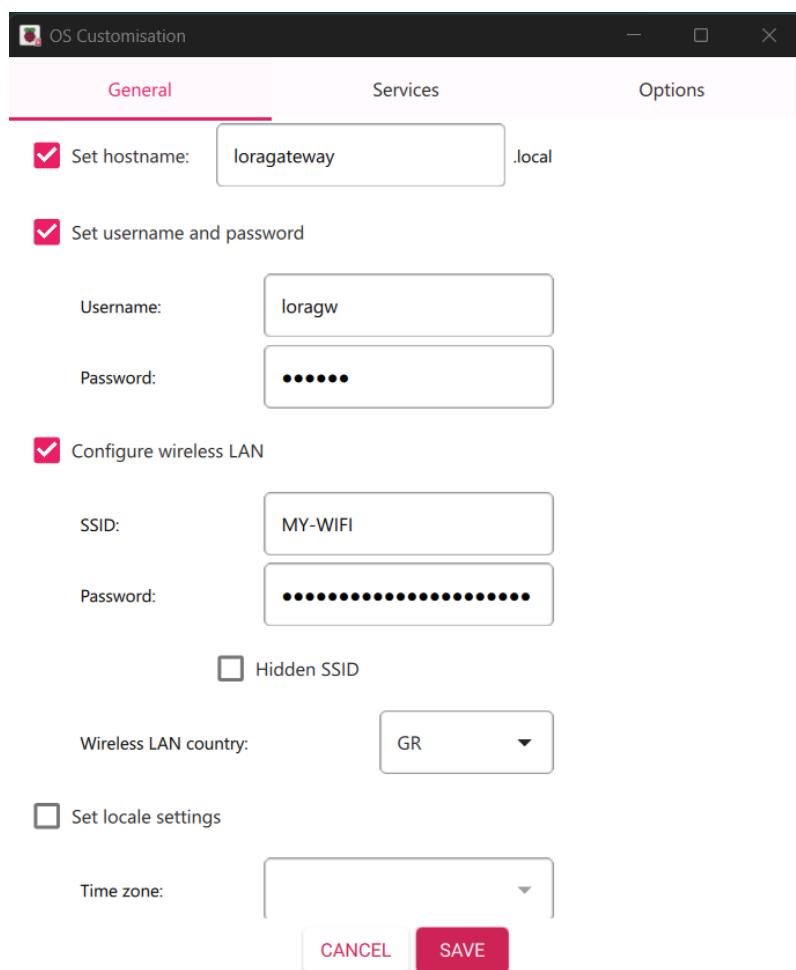
2. Επιλέγουμε **Choose Device** και από τη λίστα διαλέγουμε **Raspberry Pi 4**, που είναι το μοντέλο της συσκευής Raspberry Pi που χρησιμοποιούμε.
3. Επιλέγουμε **Choose OS** και από τη λίστα διαλέγουμε **Raspberry Pi OS Lite (64-bit)**, που είναι η έκδοση χωρίς γραφικό περιβάλλον και ενδείκνυται για servers.
4. Πατάμε **Choose Storage** και επιλέγουμε τη σωστή microSD.
5. Πατάμε **Next** και στο αναδυόμενο παράθυρο επιλέγουμε **Edit Settings** και ρυθμίζουμε:



Εικόνα 5.3: *Apply OS Customisation settings Option*.

**Στην καρτέλα General:**

- **Set Hostname:** Θέτουμε loragateway ώστε η συσκευή να είναι προσβάσιμη στο δίκτυο ως loragateway.local.
- **Set username and password:** ορίζουμε μη προεπιλεγμένα διαπιστευτήρια για λόγους ασφάλειας.
- **Configure wireless LAN (optional):** εφόσον γίνει αρχικά σύνδεση μέσω Wi-Fi, συμπληρώνουμε SSID, password και country GR. Έτσι η συσκευή μπορεί να συνδεθεί στο δίκτυό μας απευθείας, χωρίς περαιτέρω ρυθμίσεις.

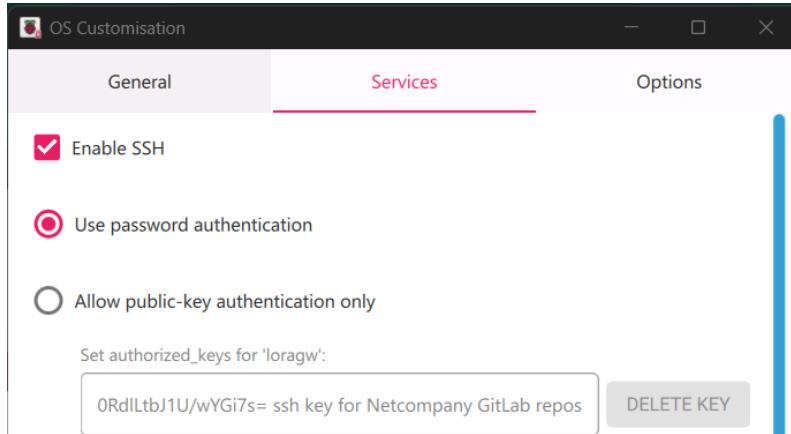


Εικόνα 5.4: General OS Customisation settings.

**Στην καρτέλα Services:**

- **Enable SSH:** ενεργοποιούμε SSH για απομακρυσμένη πρόσβαση με password.

Τέλος, πατάμε **Save** και ύστερα **Yes** στο προηγούμενο αναδυόμενο παράθυρο ώστε να εφαρμόσουμε τις ρυθμίσεις που κάναμε και να ξεκινήσει η εγγραφή της εικόνας του λειτουργικού συστήματος στην κάρτα microSD. Επιβεβαιώνουμε την επαλήθευση και αφαιρούμε με ασφάλεια το μέσο.



Εικόνα 5.5: *Services OS Customisation settings.*

6. Τοποθετούμε την microSD στο Raspberry Pi και συνδέουμε την τροφοδοσία. Το σύστημα εκκινεί σε λίγα δευτερόλεπτα.
7. Από τον υπολογιστή μας συνδεόμαστε απομακρυσμένα με SSH:

- Εντοπίζουμε τη διεύθυνση IP από το router μας (στην περίπτωσή μας έχουμε 192.168.0.100) και εκτελούμε:

```
ssh loragw@192.168.0.100
```

- Βάζουμε τον κωδικό που θέσαμε προηγουμένως για τον χρήστη loragw και συνδέόμαστε επιτυχώς.

**Σημείωση:** Επιλέξαμε να αποδώσουμε σταθερή IP στο Raspberry Pi με DHCP reservation (ανάθεση IP με βάση την MAC διεύθυνση) από το router μας ώστε η συσκευή να είναι πάντα προσβάσιμη στην ίδια διεύθυνση, κάτι κρίσιμο για σταθερά endpoints (π.χ. TTS Console, LNS/Webhook callbacks) και για κανόνες firewall/port forwarding. Έτσι αποφεύγονται διακοπές από αλλαγές IP λόγω ανανέωσης DHCP lease και απλουστεύεται η απομακρυσμένη διαχείριση (SSH), τα scripts και οι υπηρεσίες που «δείχνουν» στο gateway. Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε και domain, είτε τοπικά μέσω mDNS είτε δημόσια με καταχώριση ενός DNS A record που «δείχνει» στη διεύθυνση του gateway (ιδανικά σε συνδυασμό με Dynamic DNS ώστε να ενημερώνεται αυτόματα αν αλλάζει η IP), ώστε οι υπηρεσίες να είναι προσβάσιμες με πλήρες όνομα χώρου (FQDN).

8. Μετά την πρώτη σύνδεση, εκτελούμε βασικές ενημερώσεις συστήματος και εργαλείων συντήρησης:

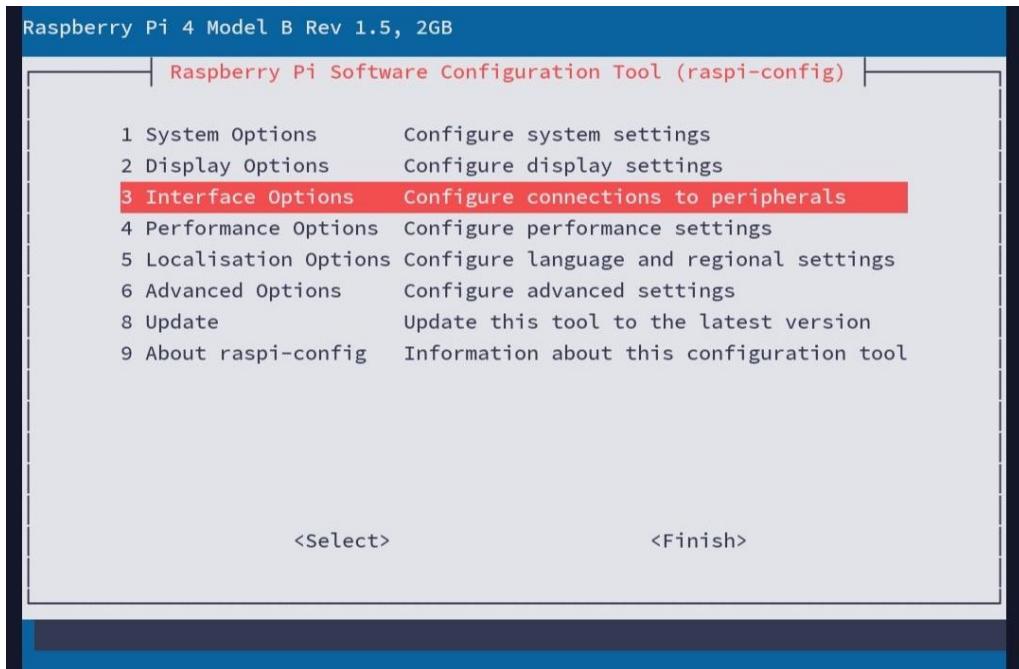
```
sudo apt update && sudo apt full-upgrade -y
sudo reboot
```

9. Μόλις ολοκληρωθεί η εγκατάσταση, για να εξασφαλιστεί ο συγχρονισμός και η σωστή επικοινωνία των δύο συσκευών, χρειάζεται να ενεργοποιηθεί η διεπαφή SPI από τις

ρυθμίσεις του Raspberry Pi. Αυτό γίνεται ανοίγοντας το εργαλείο παραμετροποίησης Raspberry Pi Software Configuration Tool με την εντολή:

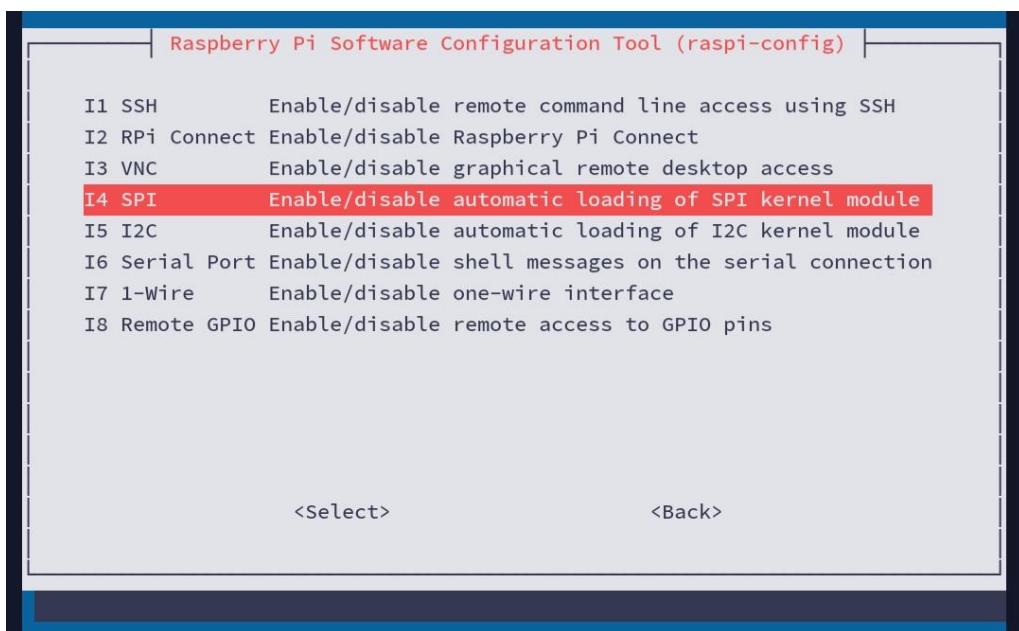
```
sudo raspi-config
```

Από το μενού που εμφανίζεται επιλέγουμε **Interface options**.



Εικόνα 5.6: Εργαλείο παραμετροποίησης λογισμικού του Raspberry Pi

Έπειτα, επιλέγουμε το **I4 SPI** και απαντάμε με yes στο αναδυόμενο παράθυρο που μας ρωτάει αν θέλουμε να ενεργοποιήσουμε την προαναφερθείσα διεπαφή:



Εικόνα 5.7: Επιλογή ενεργοποίησης διεπαφής SPI

Τέλος, πατάμε Esc στο πληκτρολόγιο ώστε να βγούμε από το εργαλείο raspi-config.

- Ολοκληρώνουμε την διαδικασία εγκατάστασης και παραμετροποίησης του λειτουργικού συστήματος εγκαθιστώντας ορισμένα απαραίτητα εργαλεία για το σήσιμο των υπηρεσιών, τρέχοντας την εντολή:

```
sudo apt-get install git gcc make
```

Με αυτά τα βήματα, το Raspberry Pi OS Lite είναι έτοιμο για την εγκατάσταση των εργαλείων και των υπηρεσιών που που θα χρειαστούν για την λειτουργία του συστήματός μας.

### 5.1.3 Εγκατάσταση Docker και Docker Compose

Για την ομαλή φιλοξενία των υπηρεσιών (The Things Stack, LoRa Basics Station και αργότερα της web εφαρμογής) στο Raspberry Pi, εγκαθίσταται το Docker Engine μαζί με το container runtime containerd και το πρόσθετο Docker Compose (νέας γενιάς ως docker compose plugin). Παρακάτω παρατίθενται τα βήματα που εκτελέστηκαν, με σύντομη επεξήγηση για κάθε εντολή.

- Κατεβάζουμε με ασφάλεια το script (εκτελέσιμο αρχείο εντολών) εγκατάστασης του Docker από τον επίσημο ιστότοπο και το αποθηκεύουμε ως get-docker.sh.

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

- Τρέχουμε το script ως root χρήστης και εγκαθιστούμε το Docker Engine, τον containerd και τα απαραίτητα πακέτα (packages) για να λειτουργήσει ομαλά το Docker.

```
sudo sh get-docker.sh
```

- Δημιουργούμε ένα Unix group με όνομα docker, ώστε οι χρήστες-μέλη του να μπορούν να εκτελούν εντολές docker χωρίς το πρόθεμα sudo (παρέχει προνόμια root χρήστη). Αν υπάρχει ήδη, η εντολή απλά αποτυγχάνει χωρίς κίνδυνο.

```
sudo groupadd docker
```

- Εντάσσουμε τον τωρινό χρήστη του περιβάλλοντος (\$USER, στην περίπτωση μας τον λοραγώ που έχουμε συνδεθεί) στο group docker για non-root χρήση του Docker.

```
sudo usermod -aG docker $USER
```

- Δημιουργούμε ένα καινούριο shell session και θέτουμε τον χρήστη μας ως μέλος του docker group χωρίς να απαιτείται πλήρης αποσύνδεση/επανασύνδεση (re-login). Εναλλακτικά, μπορούμε να κάνουμε logout/login.

```
newgrp docker
```

- Θέτουμε την υπηρεσία docker.service να εκκινεί αυτόμata σε κάθε εκκίνηση (boot) του συστήματος (systemd enable).

```
sudo systemctl enable docker.service
```

- Αντίστοιχα, ενεργοποιούμε και το containerd.service (το runtime που χρησιμοποιεί το Docker Engine).

```
sudo systemctl enable containerd.service
```

### (Προαιρετικό) Επαλήθευση εγκατάστασης

Ελέγχούμε ότι το Docker Engine και το Compose plugin είναι διαθέσιμα.

```
docker --version
docker compose version
```

```
loragw@loragateway:~$ docker --version
Docker version 28.5.0, build 887030f
loragw@loragateway:~$ docker compose version
Docker Compose version v2.40.0
```

Αν η δεύτερη εντολή δεν επιστρέψει έκδοση, εγκαθιστούμε το plugin.

```
sudo apt-get install docker-compose-plugin
```

### (Προαιρετικό) Δοκιμαστικό run.

Με την ακόλουθη εντολή τραβάμε και εκτελούμε ένα ελαφρύ δοκιμαστικό container για να επαληθεύσουμε ότι το Docker λειτουργεί σωστά και χωρίς sudo.

```
docker run --rm hello-world
```

```
loragw@loragateway:~$ docker run --rm hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

### 5.1.4 Εγκατάσταση του The Things Stack

Για την εγκατάσταση του The Things Stack αξιοποιήθηκε η έτοιμη διάταξη Docker από το ανοιχτού κώδικα αποθετήριο (repository) του GitHub xoseperez/the-things-stack-docker [56], η οποία προσφέρει ένα πλήρως παραμετροποιήσιμο compose περιβάλλον (υπηρεσίες stack, PostgreSQL, Redis, MQTT κ.ά.) και βασίζεται στο στην επίσημη εικόνα του TTS της The Things Industries. Όμοια χρησιμοποιήθηκε και αργότερα για το LoRa Basics Station το αντίστοιχο αποθετήριο του ίδιου δημιουργού. Τα συγκεκριμένα αποθετήρια απλοποιούν τη διαδικασία εγκατάστασης και παραμετροποίησης των υπηρεσιών αυτών και εξασφαλίζουν την εύρυθμη λειτουργία και επικοινωνία τους λόγω της συμβατότητάς τους. Παρακάτω καταγράφονται αναλυτικά οι ενέργειες που ακολουθήθηκαν για να στηθεί το TTS.

#### 5.1.4.1 Λήψη πηγαίου κώδικα

Εκτελούμε τις ακόλουθες εντολές στο Raspberry Pi ώστε να κλωνοποιήσουμε το αποθετήριο και να το τοποθετήσουμε σε φάκελο με όνομα TheThingsStack:

```
git clone https://github.com/xoseperez/the-things-stack-docker
mv the-things-stack-docker TheThingsStack
cd TheThingsStack/
```

```
loragw@loragateway:~/TheThingsStack $ ls
assets      CHANGELOG.md      Dockerfile      LICENSE.md
balena.yml  docker-bake.hcl   Dockerfile-lite README.md
build.sh    docker-compose.yml runner
```

Στον ριζικό κατάλογο (directory) του αποθετηρίου εντοπίζουμε το Dockerfile που καθορίζει τη βάση της εικόνας, την ενσωμάτωση των scripts του υποφακέλου runner/ και τις εντολές εκκίνησης. Το docker-compose.yml ορίζει πώς συναρμολογούνται οι υπηρεσίες (TTS, Redis, Postgres), τα volumes και οι μεταβλητές περιβάλλοντος. Το docker-bake.hcl και το build.sh υποστηρίζουν τις multi-platform builds (π.χ. arm/amd64) και απλοποιούν τη διαδικασία κατασκευής της εικόνας. Τα αρχεία .ttn-lw-stack-docker.yml.template και .ttn-lw-cli.yml.template (μέσα στο υποφάκελο runner/) χρησιμοποιούνται ως πρότυπα για τη διαμόρφωση του TTS και του CLI, με το script start να τα επεξεργάζεται. Αυτά τα στοιχεία συνεργάζονται ώστε η τελική εικόνα να «τρέχει» το επίσημη TTS image με αυτόματη διαμόρφωση, ανέλιξη πιστοποιητικών, αρχικοποίηση βάσης και δημιουργία διαχειριστών χωρίς να χρειάζεται χειροκίνητη παρέμβαση από τον χρήστη.

#### 5.1.4.2 Παραμετροποίηση docker-compose

Επεξεργαζόμαστε το docker-compose.yml ώστε να ταιριάζει στο περιβάλλον μας.

```
sudo nano docker-compose.yml
```

Αφότου ανοίξουμε το αρχείο, βλέπουμε τα εξής:

- **volumes:** Τα redis, postgres, stack-blob, stack-data ορίζονται ως named volumes για επίμονη αποθήκευση, ώστε τα δεδομένα να διατηρούνται ανεξάρτητα από επανεκκινήσεις των containers.
- **Υπηρεσία postgres:** Εκκινεί την εικόνα της βάσης PostgreSQL με αρχικοποίηση χρηστών/βάσης μέσω POSTGRES\_\* μεταβλητών. Το volume που έχει οριστεί εξασφαλίζει μόνιμη αποθήκευση, ενώ το port δένεται τοπικά (127.0.0.1:5432) για περιορισμένη πρόσβαση (μόνο localhost).
- **Υπηρεσία redis:** Εκκινεί την εικόνα της βάσης Redis με AOF (-appendonly yes) για ανθεκτικότητα. Το volume redis:/data διατηρεί την κατάσταση και η θύρα εκτίθεται μόνο σε localhost.
- **Υπηρεσία stack (The Things Stack):** Εξαρτάται από redis και postgres, προσαρτά δύο volumes για αρχεία blob και εφαρμοστικά δεδομένα, και ορίζει κρίσιμες μεταβλητές περιβάλλοντος:

*TTS\_DOMAIN*: public host/domain για Console, APIs και callbacks.

*TTN\_LW\_REDIS\_ADDRESS*: Διεύθυνση της βάσης Redis της σύνθεσης.

*TTN\_LW\_IS\_DATABASE\_URI*: Διεύθυνση της βάσης PostgreSQL του Identity Server.

*TTN\_LW\_BLOB\_LOCAL\_DIRECTORY*: τοπική αποθήκη για blobs.

- **Ports της υπηρεσίας stack:** Εκτίθενται οι απαιτούμενες θύρες για Console, API, LNS κ.α. (π.χ. 1881-1885, 8881-8887, 1700/udp). Ιδίως για LNS, η σύνδεση του LoRa Basics Station γίνεται μέσω wss στη 8887, όπως προβλέπεται από την τεκμηρίωση.

Οι αλλαγές που κάναμε στο αρχείο αυτό είναι οι ακόλουθες:

- Αφαιρέσαμε (comment out) το image configuration και ενεργοποιήσαμε (comment in) το build configuration. Έτσι, αντί να κάνουμε pull την εκάστοτε «τελευταία» εικόνα από το registry, παράγουμε το container image τοπικά από το Dockerfile, περνώντας ρητά build args (π.χ. *REMOTE\_TAG=3.32.0*, *ARCH=amd64*). Η προσέγγιση αυτή προσφέρει ακινητοποίηση έκδοσης (pinning), καλύτερο έλεγχο και αναπαραγωγιμότητα του build, καθώς και ευελιξία για μελλοντικές τροποποιήσεις στο Dockerfile (π.χ. patches, προσαρμοσμένα certs, επιπλέον tools).
- Ορίσαμε το *TTS\_DOMAIN* στη στατική IP 192.168.0.100 που δεσμεύσαμε μέσω DHCP reservation (βλ. σημείωση στο βήμα 7 της Υποενότητας 5.1.2). Με αυτόν τον τρόπο, όλες οι δημόσιες διευθύνσεις της στοίβας (Console, OAuth redirects, LNS wss:8887, MQTT/HTTP endpoints, webhooks) «δένουν» σε ένα σταθερό host. Το σταθερό endpoint απλοποιεί τις ρυθμίσεις σε gateways/εφαρμογές, διευκολύνει τη διάγνωση/ασφάλεια και προετοιμάζει τη μελλοντική μετάβαση σε DNS όνομα και TLS πιστοποιητικά χωρίς αλλαγές στις εσωτερικές ροές.

Συνεπώς, η τελική μορφή του αρχείου φαίνεται παρακάτω:

```

1  volumes:
2    redis:
3    postgres:
4    stack-blob:
5    stack-data:
6
7  services:
8    postgres:
9      image: postgres:14.3-alpine3.15
10     container_name: postgres
11     restart: unless-stopped
12     environment:
13       - POSTGRES_PASSWORD=your_postgres_password_here
14       - POSTGRES_USER=your_postgres_user_name_here
15       - POSTGRES_DB=ttn lorawan
16     volumes:
17       - 'postgres:/var/lib/postgresql/data'
18     ports:
19       - "127.0.0.1:5432:5432"
20
21   redis:
22     image: redis:7.0.0-alpine3.15
23     container_name: redis
24     command: redis-server --appendonly yes
25     restart: unless-stopped
26     volumes:
27       - 'redis:/data'
28     ports:
29       - "127.0.0.1:6379:6379"
30
31   stack:
32     #image: xoseperez/the-things-stack:latest
33     build:
34       context: .
35       dockerfile: Dockerfile
36       args:
37         ARCH: amd64
38         REMOTE_TAG: 3.32.0
39     container_name: stack
40     restart: unless-stopped
41     depends_on:
42       - redis
43       - postgres
44     volumes:
45       - 'stack-blob:/srv/ttn-lorawan/public/blob'
46       - 'stack-data:/srv/data'
47     environment:
48       TTS_DOMAIN: 192.168.0.100
49       TTN_LW_BLOB_LOCAL_DIRECTORY: /srv/ttn-lorawan/public/blob
50       TTN_LW_REDIS_ADDRESS: redis:6379
51       TTN_LW_IS_DATABASE_URI:
52       postgres://root:root@postgres:5432/ttn_lorawan?sslmode=disable
53       CLI_AUTO_LOGIN: "false"
54     labels:

```

```

54      io.balena.features.balena-api: '1'
55
56  ports:
57    - "80:1885"
58    - "443:8885"
59    - "1881:1881"
60    - "1882:1882"
61    - "1883:1883"
62    - "1884:1884"
63    - "1885:1885"
64    - "1887:1887"
65    - "8881:8881"
66    - "8882:8882"
67    - "8883:8883"
68    - "8884:8884"
69    - "8885:8885"
70    - "8887:8887"
71    - "1700:1700/udp"

```

### Εκκίνηση stack container και σύνδεση στο Console

Πλέον το The Things Stack είναι έτοιμο για εκκίνηση. Εκτελούμε:

```
docker compose up
```

```

loragw@loragateway:~/TheThingsStack $ docker compose up
[+] Running 4/4
✓ Network thethingsstack_default  Created      0.1s
✓ Container redis                Created      0.5s
✓ Container postgres              Created      0.5s
✓ Container stack                Created      0.2s

```

Μπορούμε πλέον να συνδεθούμε στο Console. Ανοίγουμε τον φυλλομετρητή (browser) και μεταβαίνουμε στη διεύθυνση <http://192.168.0.100/console>. Ο OAuth client θα μας ανακατευθύνει (redirect) στη σελίδα Login του The Things Stack Console. Για την πρώτη είσοδο χρησιμοποιούμε τον προεπιλεγμένο (default) λογαριασμό με User ID *admin* και κωδικό *changeme*. Συνιστάται άμεσα η αλλαγή κωδικού από *Home > User Settings > Change Password*, ώστε να σκληρύνουμε την ασφάλεια της εγκατάστασης.

### 5.1.5 Εγκατάσταση του LoRa Basics Station

Για τη διασύνδεση του gateway με το The Things Stack σε λειτουργία LNS (LoRa Network Server), αξιοποιήθηκε το αποθετήριο GitHub xoseperez/basicstation-docker [57], το οποίο προσφέρει έτοιμη διάταξη Docker και runner scripts για SX1301/SX1302 τύπου συγκεντρωτές. Η λογική είναι παρόμοια με του TTS: ένα container τρέχει τον Basics Station και διαβάζει καθορισμένες ρυθμίσεις από πρότυπα αρχεία (templates) του φακέλου runner, τα οποία συμπληρώνουμε με τις παραμέτρους του δικού μας gateway. Παρακάτω παρατίθενται τα ακριβή βήματα που ακολουθήθηκαν.

#### 5.1.5.1 Λήψη πηγαίου κώδικα

Κλωνοποίηση του αποθετηρίου και τοποθέτηση σε φάκελο BasicStation:

```
cd ~
git clone https://github.com/xoseperez/basicstation-docker
mv basicstation-docker BasicStation
cd BasicStation
```

```
loragw@loragateway:~/BasicStation $ ls
balena.yml      docker-bake.hcl      logo.png      tc_trust
builder         docker-compose.yml   README.md    tts-sandbox-docker-compose.yml
build.sh        Dockerfile          runner
CHANGELOG.md    Dockerfile.template ser2net.yaml
```

Στον ριζικό κατάλογο θα βρούμε αρχεία όπως Dockerfile, docker-compose.yml και τον υποφάκελο runner/. Το Dockerfile καθορίζει την εικόνα του Basics Station, το docker-compose.yml ορίζει τον τρόπο εκτέλεσης/πρόσθασης σε devices (SPI, GPIO), ενώ ο υποφάκελος runner περιέχει τα πρότυπα ρυθμίσεων (station.conf κ.λπ.), entrypoint scripts και τυχόν βοηθητικά certificates.

#### 5.1.5.2 Υπολογισμός Gateway EUI

Ο LNS ζητά Gateway EUI σε μορφή EUI-64. Συνήθως προκύπτει από τη MAC διαύθυνση του eth0 με εισαγωγή FFFE στη μέση. Μπορούμε να υπολογίζουμε, λοιπόν, το EUI από MAC διαύθυνση του eth0 και την εκτυπώνουμε στο terminal με την εντολή:

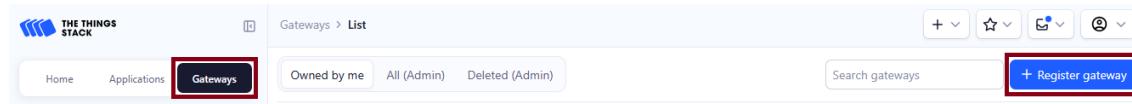
```
cat /sys/class/net/eth0/address | awk -F\: '{print toupper($1$2$3"FFFE"$4$5$6)}'
```

#### 5.1.5.3 Καταχώριση του gateway στο The Things Stack

Πριν συνδεθεί το LoRa Basics Station απαιτείται η δημιουργία εγγραφής gateway στο The Things Stack και η έκδοση κατάλληλου API key. Μέσα από το Console ορίζουμε Gateway ID, frequency plan και LNS server/route, και στη συνέχεια δημιουργούμε ένα API

key με τα ελάχιστα απαραίτητα δικαιώματα για LNS σύνδεση. Το Gateway ID και το API key θα χρησιμοποιηθούν στην παραμετροποίηση του Basic Station ώστε να επιτευχθεί η ασφαλής σύνδεση (wss) και η ανταλλαγή uplinks/downlinks με τον Network Server.

1. Από τις επιλογές αριστερά, διαλέγουμε *Gateways* και στη συνέχεια πατάμε το κουμπί **+ Register gateway** για να καταχωρίσουμε νέα συσκευή gateway:



Εικόνα 5.8: Επιλογή καταχώρησης νέου Gateway.

2. Συμπληρώνουμε το EUI που βρήκαμε προηγουμένως (βλ. Υπο-υποενότητα 5.1.5.3) και πατάμε confirm:

### Register gateway

Register your gateway to enable data traffic between nearby end devices and the network.  
Learn more in our guide on [Adding Gateways](#).

Gateway EUI ⓘ \*

45 ED 7D 64 DF D5 A6 76

Confirm

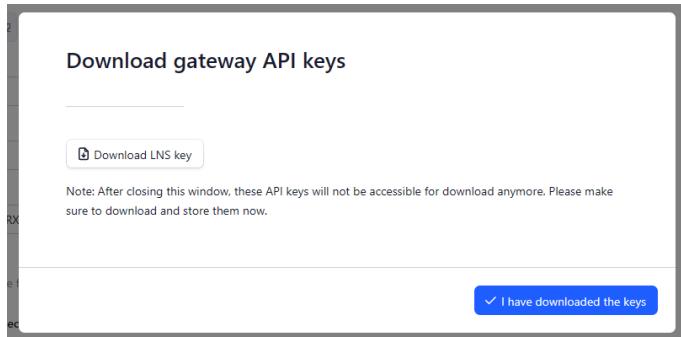
To continue, please confirm the Gateway EUI so we can determine onboarding options

Εικόνα 5.9: Εισαγωγή EUI.

3. Θέτουμε τα Gateway ID και name καθώς και το Frequency plan που επιθυμούμε. Επιπλέον ενεργοποιούμε την επιλογή *Require authenticated connection* και επιλέγουμε *Generate API key for LNS*. Τέλος πατάμε *Register Gateway*:

Εικόνα 5.10: Δημιουργία Gateway ID και name, επιλογή Frequency plan και δημιουργία κλειδιού LNS.

4. Στο αναδυόμενο παράθυρο που μας προτρέπει να αποθηκεύσουμε το gateway API key (LNS Authentication Key) επιλέγουμε Download LNS keys για να κατεβάσουμε το αρχείο με το κλειδί.



Εικόνα 5.11: Αποδήμευση API κλειδιού.

5. Τέλος, ενώ έχουμε εκκινήσει το TTS container, τρέχουμε την ακόλουθη εντολή για να εκτυπώσουμε το TLS certificate του TTS server, το οποίο θα χρειαστεί για την παραμετροποίηση του Basics Station:

```
docker exec stack get_trust_certificate
```

#### 5.1.5.4 Παραμετροποίηση docker-compose

Επεξεργαζόμαστε το docker-compose.yml του Basic Station ώστε να προσαρμοστεί στο δικό μας gateway και στον LNS του The Things Stack. Το αρχείο ορίζει ένα μόνο service (το basicstation) και παραμετροποιείται μέσω environment variables για το μοντέλο συγκεντρωτή, τη διεπαφή SPI, τη σύνδεση LNS (TC\_URI, TC\_TRUST, TC\_KEY) κ.ά.

Μόλις ανοίξουμε το αρχείο βλέπουμε τις παρακάτω ρυθμίσεις για το basicstation container:

- **image:** Επιλέγουμε την έτοιμη εικόνα aarch64 (κατάλληλη για Raspberry Pi 4). Εναλλακτικά μπορούμε να αφαιρέσουμε (comment out) την γραμμή image και να ενεργοποιήσουμε το build για να «κλειδώσουμε» REMOTE\_TAG ή να κάνουμε προσαρμογές.
- **container\_name:** Ορίζει το σταθερό όνομα του container.
- **restart: unless-stopped:** Αυτόματη επανεκκίνηση μετά από σφάλμα ή reboot. Απαραίτητο για gateway που λειτουργεί αδιάκοπα, μειώνει τις χειροκίνητες παρεμβάσεις και το χρόνο μη διαθεσιμότητας.
- **privileged: true:** Επιτρέπει πρόσβαση σε SPI/GPIO χωρίς να ορίσουμε μεμονωμένες συσκευές. Αν προτιμάμε αυστηρότερη ασφάλεια, μπορούμε να αφαιρέσουμε το privileged και να μοιραστούμε ρητά τις συσκευές (/dev/spidev0.0, /dev/gpiochip0).
- **network\_mode: host:** Το container «βλέπει» το host network stack, ώστε ο Gateway EUI (που παράγεται από MAC) να είναι σταθερός ανάμεσα σε επανεκκινήσεις. Είναι η ασφαλέστερη επιλογή για αποφυγή αλλαγών EUI.

Επεξεργαζόμαστε στη συνέχεια τις ακόλουθες μεταβλητές περιβάλλοντος ώστε το basic-station να λειτουργήσει κατάλληλα με βάση την δική μας υλοποίηση:

#### **Ρυθμίσεις συγκεντρωτή (concentrator):**

- **MODEL: "IC880A"**: Δηλώνουμε ρητά το iC880A-SPI, ώστε ο Basics Station να εφαρμόσει το σωστό design/pinout.
- **GPIO\_CHIP**: Ορίζει τον GPIO chip που θα χρησιμοποιηθεί (στο Raspberry Pi 4 τυπικά έχουμε gpiochip0).
- **RESET\_GPIO**: Η γραμμή GPIO που οδηγεί το reset του concentrator. Πρέπει να ταιριάζει με τη φυσική συνδεσμολογία (στην περίπτωσή μας 25).
- **POWER\_EN\_GPIO**: GPIO γραμμή (αν υπάρχει) που τροφοδοτεί/ενεργοποιεί το concentrator μέσω power enable pin. Αν το board δεν διαθέτει τέτοιο έλεγχο, το αφήνουμε 0 (μη χρήση).
- **POWER\_EN\_LOGIC**: Λογικό επίπεδο που «ανάβει» την τροφοδοσία στο power enable (1 για active-high, 0 για active-low). Για iC880A συνήθως δεν απαιτείται, αλλά το κρατάμε 1 ώστε, αν υπάρχει power gate, να ενεργοποιείται.
- **SPI\_SPEED**: Ταχύτητα SPI διαύλου. Θέτουμε 2000000 (2MHz) που είναι συχνά σταθερή επιλογή για iC880A. Αν δούμε SPI σφάλματα, μπορούμε να τη χαμηλώσουμε.

#### **Σύνδεση με LNS:**

- **TC\_URI**: WebSocket Secure διεύθυνση προς τον LNS του TTS. Ορίζουμε την διεύθυνση wss://192.168.0.100:8887. Η θύρα 8887 αντιστοιχεί στο LNS/Basic Station protocol.
- **TC\_TRUST**: PEM του CA/server που εμπιστεύομαστε για την TLS σύνδεση. Τοποθετούμε εδώ το trust bundle του TTS που βρήκαμε στο βήμα 5 της Υπο-υποενότητας 5.1.5.3 ώστε να ολοκληρώνεται το TLS handshake.
- **TLS\_SNI: false**: Απενεργοποίηση SNI ελέγχου, χρήσιμη όταν χρησιμοποιούμε self-signed certs ή IP αντί για FQDN.
- **SERVER**: Βοηθητική μεταβλητή για runner scripts του image, ίση με τη διεύθυνση του TTS (δεν αντικαθιστά το TC\_URI).
- **TC\_KEY: Gateway LNS API key** από το TTS (δικαίωμα Link as Gateway). Πρέπει να ανήκει στο συγκεκριμένο gateway που καταχωρίσαμε και να είναι ενεργό/μη ανακλημένο. Θέτουμε εδώ το κλειδί που πήραμε στο βήμα 4 της Υπο-υποενότητας 5.1.5.3.

Επομένως η τελική μορφή του docker-compose.yml είναι:

```
1  services:
2    basicstation:
3
4      image: xoseperez/basicstation:aarch64-latest
5      #build:
6      # context: .
7      # args:
8      #   ARCH: aarch64
9      #   REMOTE_TAG: v2.0.6
10     #   VARIANT: std
11
12     container_name: basicstation
13     restart: unless-stopped
14     privileged: true          # set this to true or define the required devices to share
15       with the container under the `devices` tag below
16     network_mode: host        # required to read host interface MAC instead of virtual
17       one, you don't need this if you set the GATEWAY_EUI manually
18
19     environment:
20
21       # To select your concentrator you have 3 options:
22       # Option 1: set the MODEL to the device model number (i.e. RAK7371)
23       # Option 2: set the MODEL to the concentrator model number (i.e. RAK5146, WM1302,
24       R11E-LR8...)
25       # Option 2: set the MODEL to the concentrator chip (i.e. SX1303)
26       MODEL: "IC880A"
27
28       # GPIO to reset SPI concentrators (these are the defaults)
29       GPIO_CHIP: "gpiochip0" # set to "gpiochip4" by default for Raspberry Pi 5
30       RESET_GPIO: 25
31       POWER_EN_GPIO: 0
32       POWER_EN_LOGIC: 1
33
34       # Problems with an SPI concentrator are sometimes related to the bus speed,
35       # you can set a different one this way
36       SPI_SPEED: 2000000
37
38       # Server name indication (SNI) check, defaults to true.
39       # Use it only to connect to private LNS/CUPS servers with self-signed certificates
40       TLS_SNI: false
41
42       # -----
43       # LNS Protocol
44       # -----
45
46       # Using LNS protocol: define a custom LNS server
47       TC_URI: "wss://192.168.0.100:8887"
48
49       # Using LNS protocol: if you use a server other than TTN, you will have to
50       # provide the certificate for that server.
51       TC_TRUST: "-----BEGIN CERTIFICATE-----MIID9TCCAt2gAwIBAgIUF41fFXcGDe13+lCrpAfqb8WP
52       FW4wDQYJKoZIhvcNAQELBQAwZTELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUNhdGFsdW55YTESBAGA1UEBx
53       MJQmFyY2Vsb25hMRYwFAYDVQQKEw1UVE4gQ2F0YWx1bnlhMRYwFAYDVQQDEw0xOTIuMTY4LjAuMTAwMB4X
54       DTI0MTEwODAwNDkwMFoXTDI1MTEwODAwNDkwMFowZTELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUNhdGFsdW
```

```

51      55YTESBAGA1UEBxMJQmFyY2Vsb25hMRYwFAYDVQQKEw1UVE4gQ2F0YWx1bn1hMRYwFAYDVQQDEw0x0TIu
52      MTY4LjAuMTAwMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBcgKCAQEAsNgToNXbBk5WYvCTz0Vy8NZnFJ
53      eFIyON8TU900dMQ50AY1nNSVsAW0ijUuft9tVnmJ1fdpu2oo3dOnQ/wdzw10EFwLu1h5XI9c3MER49Tngr
54      N83kp/4xkpsNt+52hiS780QgwBqNd9yg5hL9GAgIxet6UAzIQvcuLFFdfLvoOTcG6HNF1+YHR/Yk4AHn9
55      Rs7FtF0HDVigkPXGo6X6HzjS7uPDGSxiK3h0qaNxix+Ly4Nn7RDzHdeEJ9pt0JxEi8z3atNqQJ2zhZWpYy
56      L9eL/u5JMyUUfXjdootqQNAPfepsBLS3RZMNiwcvnmtUhJm/OM2V4TtnVNPPFBphzt9uwIDAQABo4GcMI
57      GZMA4GA1UdDwEB/wQEAwIFoADbgNVHSUEfjAUBggrBgfEFBQcDAQYIKwYBBQUHAWiWDAYDVR0TAQH/BAIw
58      ADAdBgNVHQ4EFgQUhwmGZcu73IE1pxpxMNgQ/+9w0u4wHwYDVR0jbBgwFoAUhxUDzmo700iHwReNRshBjX
59      zuZRYwGgYDVR0RBMMwEYIJbG9jYWxob3N0hwTAqABkMA0GCSqGSIb3DQEBCwUAA4IBAC7DJVThScd5FkL
60      5k4xEsTw0KeuSoVyxmpuYUn5KKQBSiOYvlm8VSA+O3uXknIKvel17JIsKQIKoXXoAMxbKYhZ1oFhdKT
61      apVu8Bn1GPRKM+X89eBs0DXZLWxVEZQorfQvP5z0gXrL9AV3t6qb1UMKC4VZgk6AQZZgRkvIk57HcaQ80
62      qF7K7A8fH4YXH+s/bw++9hkdmPxSJYxvWJGwgujX7BnNujYt670aPCx+uBqYQU6L/bJtEpALZsIWSG1+5
63      sMgM7fZC+yzKUSw65K3w8ge+Zb04sjQP1T4YtMbqZdm9njOoVBKA50p6UZViuxygum2fSVaGCgiZtNzhrZ
64      -----END CERTIFICATE-----"
65
66      # Finally, you need to provide the key for the gateway,
67      # this can either be the client key for the client certificate above or
68      # the API key provided by the LNS (if no client certificate)
69      SERVER: "192.168.0.100"
70      TC_KEY: "NNSXS.07YTVAWDVA0275AHJZYNGFSEPAATRW4D6K72PSI.FM6CU4E7Y2ERZI7U4QTGRL6WG4Q
71      502TDQA7M00UA55ZK5NYD3KRA"

```

### Εκκίνηση Basic Station και έλεγχος λειτουργίας

Μετά τις ρυθμίσεις, εκκινούμε το container (έχοντας ήδη εκκινήσει το container του TTS):

```
docker compose up
```

```

loragw@loragateway:~/BasicStation $ docker compose up
[+] Running 4/4
  ✓ Container basicstation Created          0.3s

```

Στα logs του container αναμένουμε επιτυχές LNS connect και αποδοχή από τον Gateway Server:

```

loragw@loragateway:~/BasicStation $ docker logs basicstation
...
2025-10-19 14:44:00.891 [TCE:INFO] Infos: d83a:ddff:fea4:81a7 muxs-::0
wss://192.168.0.100:8887/traffic/eui-D83ADDFFE481A7
2025-09-19 14:44:01.028 [TCE:VERB] Connected to MUXS.
...

```

Στο TTS Console το gateway θα εμφανιστεί **connected**. Από εδώ και πέρα, τα uplinks των μετρητών θα δρομολογούνται στον Network Server.

The screenshot shows the 'Gateway overview' page for a specific gateway. The left sidebar lists other gateways. The main area displays 'General information' and 'Network settings'. The 'Gateway status' section is highlighted with a red box, showing '30 day uptime' and 'Connection stats' which indicate no uplinks or downlinks yet.

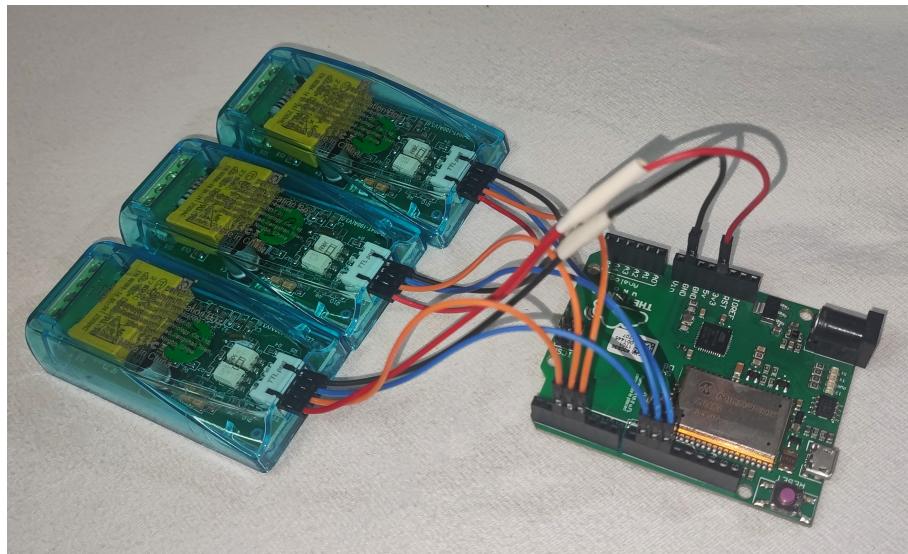
Εικόνα 5.12: *Gateway Overview and status στο TTS Console.*

Με τα παραπάνω, το LoRa Basics Station τρέχει ως container στο Raspberry Pi και συνδέεται επιτυχώς στον Gateway Server του The Things Stack, ολοκληρώνοντας το σκέλος του gateway. Στα επόμενα βήματα, οι τριφασικοί μετρητές θα αρχίσουν να στέλνουν uplinks μέσω του gateway προς το TTS.

## 5.2 Προετοιμασία τριφασικού μετρητή

### 5.2.1 Διασύνδεση Hardware

Όπως αναλύθηκε στην Ενότητα 4.3, συνδέουμε την πλακέτα The Things Uno με τα τρία PZEM-004T σύμφωνα με την Εικόνα 4.3 και τους Πίνακες 4.8, 4.9 και 4.10:



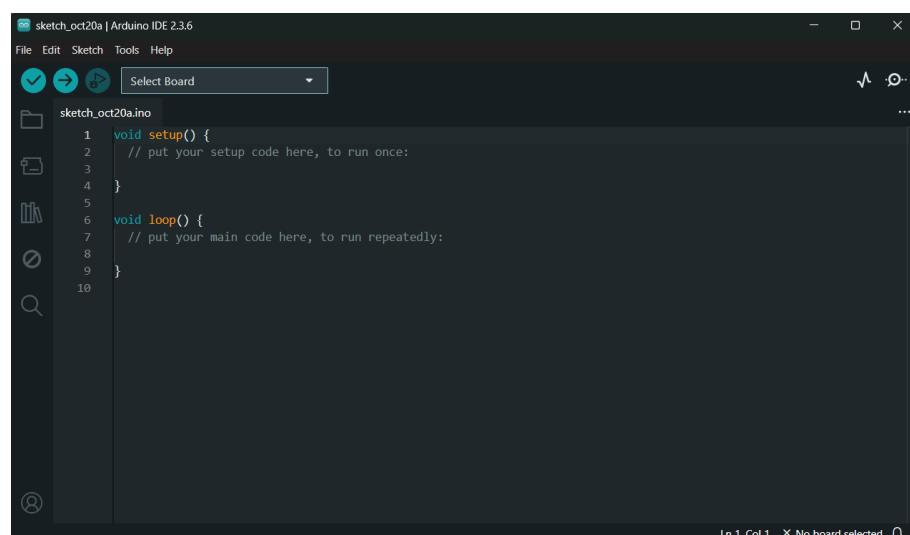
Εικόνα 5.13: *Τελική συνδεσμολογία των εξαρτημάτων του τριφασικού μετρητή.*

## 5.2.2 Προγραμματισμός του The Things Uno

Ακολουθούμε τις επίσημες οδηγίες της The Things Industries για τη σύνδεση της πλακέτας The Things Uno με το TTS [39]:

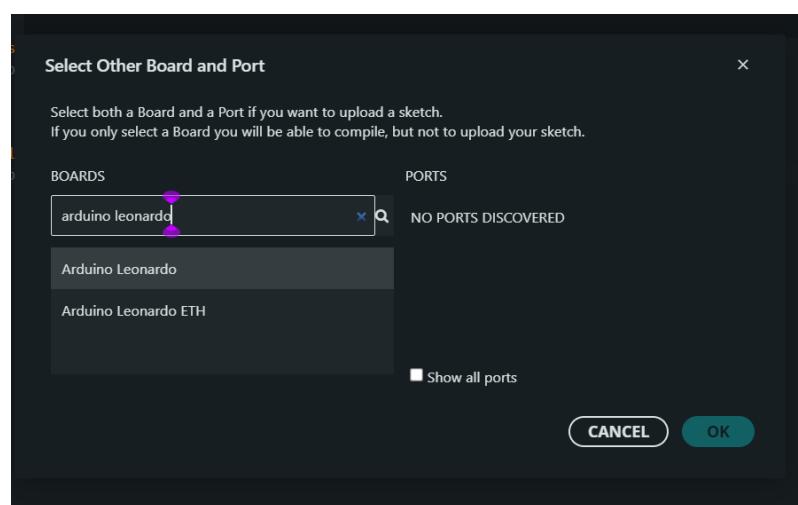
### 5.2.2.1 Εγκατάσταση και ρύθμιση Arduino IDE

Προκειμένου να προγραμματίσουμε την πλακέτα The Things Uno του τριφασικού μετρητή θα χρειαστεί να κατεβάσουμε αρχικά το λογισμικό Arduino IDE στον προσωπικό μας υπολογιστή. Περιλαμβάνει έναν επεξεργαστή κειμένου, έναν μεταγλωττιστή (compiler) και ένα σειριακό τερματικό, καθιστώντας εύκολη τη σύνταξη, τη μεταγλώττιση και τη φόρτωση προγραμμάτων στον μικροελεγκτή πλακετών Arduino. Μόλις εγκαταστήσουμε το Arduino IDE το εκκινούμε και αυτόματα ανοίγει ένα καινούριο sketch:



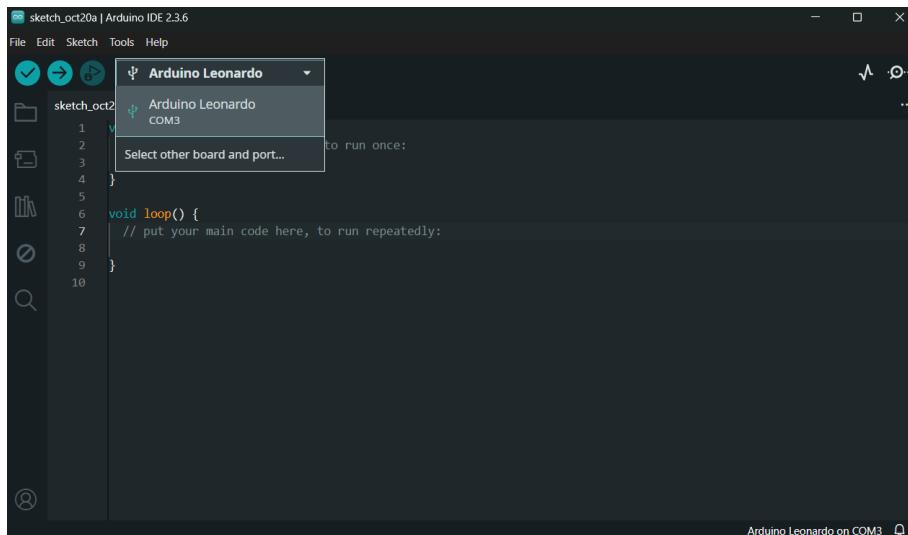
Εικόνα 5.14: Λογισμικό Arduino IDE.

Στη συνέχεια πατάμε *Select Board* και στην αναζήτηση που εμφανίζει συμπληρώνουμε *Arduino Leonardo*, το επιλέγουμε και πατάμε *ok* για να καταχωρίσουμε την επιλογή μας.



Εικόνα 5.15: Αναζήτηση και επιλογή πλακέτας Arduino Leonardo.

Τέλος, συνδέουμε το The Things Uno στον υπολογιστή μας και αυτόματα εμφανίζεται η επιλογή της θύρας COM# που έγινε η σύνδεση. Την διαλέγουμε και επιβεβαιώνουμε ότι κάτω δεξιά αναγράφεται *Arduino Leonardo on COM#*.

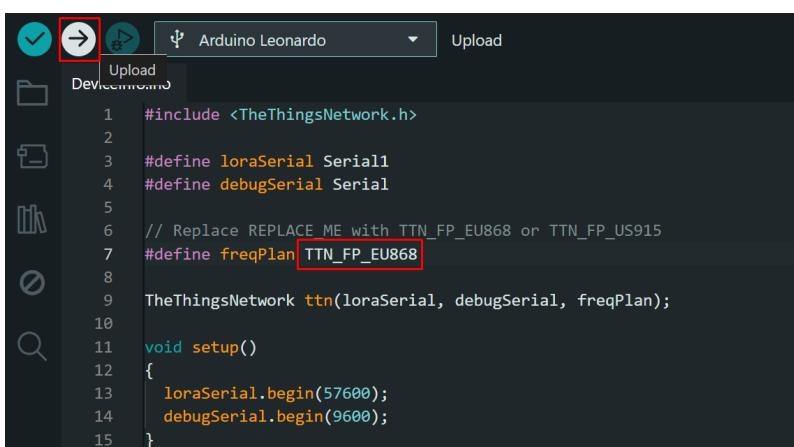


Εικόνα 5.16: Επιλογή Arduino Leonardo στη θύρα COM3 όπου έχει συνδεθεί το The Things Uno.

Μπορούμε τώρα να ξεκινήσουμε να γράφουμε τον κώδικα (σε γλώσσα προγραμματισμού C++) για τον προγραμματισμό του The Things Uno και να τον «ανεβάσουμε» (upload) στην πλακέτα.

### Σημείωση

Σε αυτό το σημείο, προκειμένου να βρούμε ορισμένες σημαντικές πληροφορίες για την συσκευή TTU μπορούμε να τρέξουμε το sketch που βρίσκεται στην διαδρομή *File > Examples > TheThingsNetwork > Deviceinfo*. Αφού το επιλέξουμε, θέτουμε στην μεταβλητή **freqPlan** την τιμή *TTN\_FP\_EU868* πατάμε το κουμπί του Upload:



Εικόνα 5.17: Κάνουμε upload το sketch στο TTU.

Από την διαδρομή *Tools > Serial Monitor* ανοίγουμε το Serial Monitor και βλέπουμε την έξοδο του TTU με τις πληροφορίες της συσκευής:

```

EUI: 0004A30B001BDB64
Battery: 3223
AppEUI: 0000000000000000
DevEUI: 0004A30B001BDB64
Data Rate: 5
RX Delay 1: 1000
RX Delay 2: 2000

Use the EUI to register the device for OTAA
-----
```

Εικόνα 5.18: Έξοδος του sketch Deviceinfo στο Serial Monitor με τις πληροφορίες της συσκευής.

### 5.2.2.2 Δομή κώδικα και ροή λειτουργίας

Ακολουθεί ο κώδικας που χρησιμοποιήσαμε και στη συνέχεια η επεξήγησή του:

```

1 #include <TheThingsNetwork.h>
2 #include <PZEM004Tv30.h>
3 #include <SoftwareSerial.h>
4
5 #define loraS Serial1
6 #define debugS Serial
7 #define freqPlan TTN_FP_EU868
8 #define NUM_PZEMS 3
9
10 TheThingsNetwork ttn(loraS, debugS, freqPlan);
11 SoftwareSerial pzemSWSerial1(9, 2); // RX, TX for PZEM 1
12 SoftwareSerial pzemSWSerial2(10, 3); // PZEM 2
13 SoftwareSerial pzemSWSerial3(11, 4); // PZEM 3
14 PZEM004Tv30 pzmets[NUM_PZEMS];
15
16 // TTN Configuration
17 const char *appEui = "0000000000000000";
18 const char *appKey = "EE5E017B3C82DEE19DD069432C9C445";
19 // Energy reset configuration
20 unsigned long lastResetTime = 0;
21 const unsigned long resetInterval = 15 * 60000; // 15 minutes in milliseconds
22 // Sentinel for "invalid" fields
23 static const uint16_t INVALID16 = 0xFFFF;
24 static const uint32_t INVALID32 = 0xFFFFFFFF;
25
26 // Helper: encoding with scale into uint16_t (with check if NaN/Inf)
27 inline uint16_t encodeScaled(float v, float scale) {
28     // NaN or Inf or negative -> invalid
29     if (!isfinite(v) || isnan(v) || v < 0.0f || scale <= 0.0f) return INVALID16;
30     long val = lroundf(v * scale); // rounding
31     if (val > 0xFFFF) return INVALID16; // out of range -> invalid
32     return (uint16_t)val;
33 }
34
35 // Helper: write big-endian 16-bit
```

```

36 inline void putBE(uint8_t *buf, int idx, uint16_t u) {
37     buf[idx] = (uint8_t)(u >> 8);
38     buf[idx + 1] = (uint8_t)(u & 0xFF);
39 }
40
41 // Helper: write big-endian 32-bit
42 inline void putBE32(uint8_t *buf, int idx, uint32_t u) {
43     buf[idx + 0] = (uint8_t)((u >> 24) & 0xFF);
44     buf[idx + 1] = (uint8_t)((u >> 16) & 0xFF);
45     buf[idx + 2] = (uint8_t)((u >> 8) & 0xFF);
46     buf[idx + 3] = (uint8_t)((u >> 0) & 0xFF);
47 }
48
49 // Helper: Send raw IEEE-754 float in big-endian
50 inline void putFloatBE(uint8_t *buf, int idx, float energy) {
51     // NaN or Inf or negative -> sentinel
52     if (!isfinite(energy) || isnan(v) || v < 0.0f || scale <= 0.0f) {
53         putBE32(buf, idx, INVALID32);
54         return;
55     }
56     union { float f; uint32_t u; } v;
57     v.f = energy;                                // raw IEEE-754 bits
58     putBE32(buf, idx, v.u);
59 }
60
61 void setup() {
62     debugS.begin(9600);
63     loraS.begin(57600);
64     // TTN setup
65     debugS.println("## JOIN");
66     ttn.join(appEui, appKey);
67     while (!debugS) { ; }    // Wait for serial to be ready
68     debugS.println("## STATUS");
69     ttn.showStatus();
70     // Initialize PZEMs
71     pzems[0] = PZEM004Tv30(pzemSWSerial1);
72     pzems[1] = PZEM004Tv30(pzemSWSerial2);
73     pzems[2] = PZEM004Tv30(pzemSWSerial3);
74     // Initialize reset timer
75     lastResetTime = millis();
76 }
77
78 void loop() {
79     // Check if it's time to reset energy counters
80     if (millis() - lastResetTime >= resetInterval) {
81         resetEnergyCounters();
82         lastResetTime = millis();
83         debugS.println("Energy counters reset");
84     }
85
86     // Buffer for LoRaWAN payload 14 bytes per PZEM (V,C,P,f,PF = 2 each, E = 4, 42 total)
87     uint8_t data[NUM_PZEMS * 14];
88
89     // Read and process data for each PZEM
90     for (int i = 0; i < NUM_PZEMS; i++) {

```

```

91     float voltage = pzems[i].voltage();
92     float current = pzems[i].current();
93     float power = pzems[i].power();
94     float energy = pzems[i].energy();
95     float frequency = pzems[i].frequency();
96     float pf = pzems[i].pf();
97     // Validate and encode readings
98     int index = i * 14;
99     uint16_t vEnc = encodeScaled(voltage, 10);
100    uint16_t cEnc = encodeScaled(current, 100);
101    uint16_t pEnc = encodeScaled(power, 10);
102    uint16_t fEnc = encodeScaled(frequency, 10);
103    uint16_t pfEnc = encodeScaled(pf, 100);
104
105    debugS.print("PZEM "); debugS.print(i + 1); debugS.println(" Readings:");
106    debugS.print("Voltage: "); debugS.print(voltage); debugS.println(" V");
107    debugS.print("Current: "); debugS.print(current, 3); debugS.println(" A");
108    debugS.print("Power: "); debugS.print(power); debugS.println(" W");
109    debugS.print("Energy: "); debugS.print(energy, 4); debugS.println(" kWh");
110    debugS.print("Frequency: "); debugS.print(frequency); debugS.println(" Hz");
111    debugS.print("Power Factor: "); debugS.println(pf); debugS.println();
112
113    // Prepare data for LoRaWAN: V(2),I(2),P(2),E(4 float),f(2),PF(2)->14 bytes/phase
114    putBE(data, index + 0, vEnc);
115    putBE(data, index + 2, cEnc);
116    putBE(data, index + 4, pEnc);
117    putFloatBE(data, index + 6, energy); // <-- 4 bytes raw float (big-endian)
118    putBE(data, index + 10, fEnc);
119    putBE(data, index + 12, pfEnc);
120 }
121
122 debugS.println("-- Sending data to TTN");
123 ttn.sendBytes(data, sizeof(data)); // Send data to TTN
124 debugS.println("-- Data sent!");
125 delay(5*60000); // 5 minutes between readings
126 }
127
128 void resetEnergyCounters() {
129     for (int i = 0; i < NUM_PZEMS; i++) {
130         pzems[i].resetEnergy();
131         debugS.print("Reset energy for PZEM "); debugS.println(i + 1);
132     }
133 }
```

## Αναλυτική επεξήγηση κώδικα και ροής λειτουργίας

### Σύνδεση με το The Things Network:

- Συμπεριλαμβάνουμε τη βιβλιοθήκη **TheThingsNetwork.h** της The Things Network για ενεργοποίηση OTAA, αποστολή uplinks κ.λπ. Προκειμένου να αναγνωρίσει ο compiler την βιβλιοθήκη, την εγκαθιστούμε ακολουθώντας την διαδρομή *Sketch > Include Library > Manage Libraries*, αναζητώντας το όνομα της και πατώντας *install*.

- Δημιουργούμε τις σταθερές **appEui** και **appKey**, οι οποίες αποτελούν τα OTAA credentials της συσκευής. Η τιμές τους προκύπτουν κατά την καταχώριση του TTU στο TTS με την διαδικασία που περιγράφεται στην Υπο-υποενότητα 5.2.3.2, στο βήμα 3,
- **#define loraS Serial1:** Η The Things Uno (βασισμένη σε ATmega32u4/Leonardo) χρησιμοποιεί τη **Serial1** για σύνδεση με το LoRa module (π.χ. RN2483/RN2903), αφήνοντας την **Serial** για debug προς τον USB.
- **#define debugS Serial:** USB CDC για μηνύματα κατάστασης/εντοπισμό σφαλμάτων.
- **#define freqPlan TTN\_FP\_EU868:** Επιλέγουμε frequency plan EU868.
- **TheThingsNetwork ttn(loraS, debugS, freqPlan):** Δημιουργούμε TTN stack αντικείμενο με hardware UART για LoRa και USB UART για logs.

#### Διαμόρφωση PZEM-004T και SoftwareSerial:

- Συμπεριλαμβάνουμε τις βιβλιοθήκες **SoftwareSerial.h** και **PZEM004Tv30.h** (την κατεβάζουμε και αυτή από το μενού *Manage Libraries*) για UART επικοινωνία με τα PZEM (v3.0) μέσω software serial.
- **#define NUM\_PZEMS 3:** Τρεις μετρητές (μία φάση ανά PZEM).
- **SoftwareSerial pzemSWSerial1(9, 2), pzemSWSerial2(10, 3), pzemSWSerial3(11, 4):** Τρία ανεξάρτητα software UARTs. Η μορφή είναι (RX, TX) ως προς το Arduino, τα RX 9, 10, 11 πηγαίνουν στα TX των PZEM, ενώ τα TX 2, 3, 4 στα RX τους (σταυρωτά).
- **PZEM004Tv30 pzem[**NUM\_PZEMS**]:** Πίνακας driver για τα τρία PZEM.

#### Βοηθητικές ρουτίνες κωδικοποίησης (helpers):

- **INVALID16 = 0xFFFF, INVALID32 = 0xFFFFFFFF:** Ειδικές sentinel τιμές που σηματοδοτούν «άκυρο/μη διαθέσιμο» μέγεθος (π.χ. όταν μια μέτρηση είναι NaN ή εκτός ορίων).
- **encodeScaled(float v, float scale):** Κλιμακώνει την τιμή v με scale, κάνει στρογγυλοποίηση (round) και την επιστρέφει ως uint16. Αν η είσοδος δεν είναι πεπερασμένη (NaN/Inf) ή υπερχειλίζει το εύρος του uint16, επιστρέφει INVALID16.
- **putBE, putBE32:** Γράφουν uint16/uint32 (2 bytes) αντίστοιχα σε διάταξη big-endian (πρώτα το high byte), ώστε ο αποκωδικοποιητής να ξέρει με σαφήνεια τη σειρά των bytes.
- **putFloatBE:** Τοποθετεί IEEE-754 float32 (4 bytes) σε big-endian. Αν η τιμή είναι NaN/Inf, γράφεται το INVALID32 ως σαφής δείκτης σφάλματος.

### Επαναφορά μετρητών ενέργειας και ρυθμός δειγματοληψίας:

- **resetInterval = 15 \* 60000:** Ορίζουμε auto-reset ενέργειας ανά 15 λεπτά μέσω resetEnergyCounters().
- **Παρατήρηση:** Η ενέργεια (kWh) που δίνουν τα PZEM είναι **αθροιστική** και κανονικά δεν μηδενίζεται περιοδικά. Το **Αυτόματο reset ανά 15'** διακόπτει τη συνέχεια των μετρήσεων και δυσκολεύει τον υπολογισμό των ημερήσιων/μηνιαίων αθροισμάτων. Για «βάρδιες» ή χρονικά παράθυρα, θα προτιμούσαμε **λογικό μηδενισμό στην εφαρμογή**, δηλαδή να κρατάμε μία βάση (baseline) στην αρχή του παραθύρου και να υπολογίζουμε τη διαφορά  $E_{now} - E_{baseline}$ . Πραγματικό reset στη συσκευή έχουμε μόνο κατ' εξαίρεση, π.χ. με σπάνιο downlink-trigger όταν το απαιτεί η λειτουργία.
- **delay(5\*60000):** Αποστολή ανά 5 λεπτά (φιλική προς duty cycle/FUP limits, βλ. Υπο-υποενότητα 2.4.5.2).

### Συνάρτηση setup() - Αρχικοποίηση:

- **debugS.begin(9600), loraS.begin(57600):** Θέτουμε τα baud rates.
- **ttn.join(appEui, appKey):** OTAA join. Ο κώδικας μπλοκάρει μέχρι να ολοκληρωθεί το join.
- **pzem[i] = PZEM004Tv30(pzemSWSerialX):** Binding κάθε driver με το αντίστοιχο SoftwareSerial.

### Συνάρτηση loop() - Ανάγνωση, κωδικοποίηση και αποστολή:

Ορίζουμε το TTU να εκτελεί επαναληπτικά τα ακόλουθα βήματα:

1. **Αυτόματος μηδενισμός ενέργειας:** Κάθε 15 λεπτά ελέγχεται το χρονόμετρο και, αν έχει λήξει, καλείται η resetEnergyCounters() για όλους τους PZEM και επαναφέρεται ο μετρητής (lastResetTime = millis()).
2. **Ανάγνωση μεγεθών:** Για κάθε PZEM διαθέτονται voltage(), current(), power(), energy(), frequency(), pf(). Όλες οι τιμές είναι float ή NaN σε αποτυχία. Παράλληλα τυπώνονται στο debug για έλεγχο.
3. **Έλεγχος εγκυρότητας και κλιμάκωση:** Οι αριθμητικές τιμές κωδικοποιούνται επιτόπου:
  - Για τα πεδία uint16 (όλα τα μεγέθη εκτός της ενέργειας) χρησιμοποιείται η encodeScaled(v, scale), η οποία:
    - (α) απορρίπτει NaN/±Inf ή υπερχειλίσεις επιστρέφοντας INVALID16 = 0xFFFF,
    - (β') στρογγυλοποιεί (round) το v \* scale και επιστρέφει uint16.
  - Για την ενέργεια αποστέλλεται το ακατέργαστο float32 σε big-endian με putFloatBE(), που βάζει INVALID32 = 0xFFFFFFFF αν το float είναι άκυρο.

- 4. Δημιουργία payload:** Για κάθε φάση πακετάρονται 14 bytes με σταθερή διάταξη (τάση - 2, ρεύμα - 2, ισχύς - 2, ενέργεια - 4, συχνότητα - 2, pf - 2), σε big-endian. Με τρεις φάσεις το συνολικό μήκος είναι 42 bytes (< 51 bytes), γεγονός που επιτρέπει το σύστημα να λειτουργεί και σε χαμηλά data rates / υψηλά spreading factors (βλ. Πίνακα 2.10). Οι κλίμακες και τα εύρη τιμών για κάθε μέτρηση συνοψίζονται στον Πίνακα 5.1.

**Σημείωση:** Αν στέλναμε κάθε μέτρηση ως πλήρες float32 (4 bytes), θα είχαμε 6 μεγέθη × 4 bytes = 24 bytes ανά PZEM και 72 bytes συνολικά για τρεις φάσεις (δηλαδή πάνω από το «οτενό» όριο 51 bytes που ισχύει στα χαμηλά data rates της EU868, βλ. Πίνακα 2.10). Αυτό θα μας «ελείδωνε» πρακτικά σε υψηλότερα DR (π.χ. DR3/SF9 και πάνω), μειώνοντας την εμβέλεια. Για να διατηρήσουμε συμβατότητα με τα χαμηλά DR/υψηλά SF και να κρατήσουμε το payload στα ≤51 bytes, εφαρμόζουμε «συμπίεση» με κλιμάκωση στα uint16 για τάση/ρεύμα/ισχύ/συχνότητα/pf (2 bytes έκαστο) και κρατάμε μόνο την ενέργεια ως float32 (4 bytes) για να μην χάνεται η δυναμική/ακρίβεια στα αθροιστικά kWh. Έτσι προκύπτουν 14 bytes ανά φάση και 42 bytes σύνολο (μέσα στο όριο των 51 bytes). Εναλλακτικά, αν χρειάζονταν όλες οι τιμές ως float32, θα μπορούσαμε να στέλνουμε δύο διαδοχικά uplinks (π.χ. 36+36 bytes) με κατάλληλο χρονισμό και πάντα σεβόμενοι τα duty cycles. Εντούτοις, αυτή η λύση αυξάνει τον χρόνο στον αέρα (ToA) και την κατανάλωση και είναι πιο ευαίσθητη σε απώλειες πακέτων.

- 5. Αποστολή:** Το uplink στέλνεται με την συνάρτηση `ttn.sendBytes(data, sizeof(data))` και ακολουθεί καθυστέρηση 5 λεπτών για την επόμενη αποστολή (duty cycle επιτρεπτό διάστημα).

Offset	Πεδίο	Τύπος/Κλίμακα	Ακρίβεια	Εύρος τιμών
0-1	Τάση	uint16, V × 10	0.1 V	0...6553.5V
2-3	Ρεύμα	uint16, A × 100	0.01 A	0...655.35A
4-5	Ισχύς	uint16, W × 10	0.1 W	0...6553.5W
6-9	Ενέργεια	float32 (raw, BE)	kWh, IEEE-754	0...~3.4×10 <sup>38</sup> kWh
10-11	Συχνότητα	uint16, Hz × 10	0.1 Hz	0...6553.5Hz
12-13	Συν. ισχύος	uint16, × 100	0.01	0.00...655.35

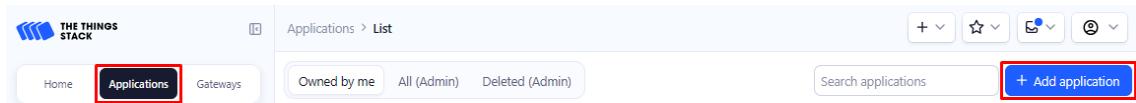
Πίνακας 5.1: Διάταξη payload ανά φάση (14 bytes/phase, σύνολο 42 bytes για 3 PZEM).

### 5.2.3 Καταχώριση του The Things Uno στο The Things Stack

#### 5.2.3.1 Δημιουργία Application

Για να συνδεθεί το The Things Uno στο The Things Stack, πρέπει πρώτα να το καταχωρίσουμε ως end device σε μία εφαρμογή (application) στο The Things Stack Console:

1. Από τις επιλογές αριστερά, διαλέγουμε *Applications* και στη συνέχεια πατάμε το κουμπί + *Add application* ώστε να προσθέσουμε νέο Application:



Εικόνα 5.19: Επιλογή προσδήκης νέου Application.

2. Συμπληρώνουμε τα Application ID και Application name που επιθυμούμε και πατάμε *Create application*. Στη συνέχεια δημιουργείται το Application και ανακατευθυνόμαστε στο Application Overview αυτουνού:

Application ID\*  
ntus-3-phase-power-meters

Application name  
3-Phase Power Meters

Description  
Description for my new application

Optional application description; can also be used to save notes about the application

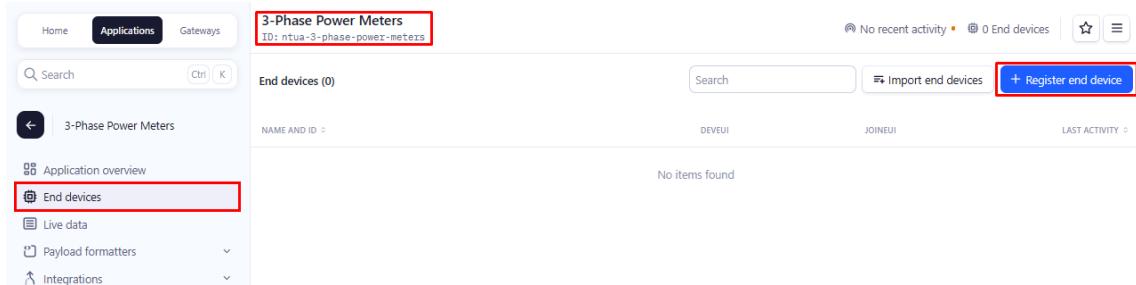
Create application

Εικόνα 5.20: Ορισμός Application ID και Application name.

### 5.2.3.2 Εγγραφή TTU στο Application

Έχοντας δημιουργήσει την εφαρμογή, προχωράμε στην καταχώριση του The Things Uno ως end device προκειμένου να μπορεί να το αναγνωρίσει ο The Things Stack Network server:

1. Ξεκινώντας από το *Application Overview* της εφαρμογής που δημιουργήσαμε, επιλέγουμε από το μενού αριστερά *end devices* και πατάμε *+ Register end device*:



Εικόνα 5.21: Επιλογή εγγραφής end device.

2. Στη συνέχεια, μας ζητείται να επιλέξουμε *End device type (Input Method)*. Επιλέγουμε *Select the end device in the LoRaWAN Device Repository* και ύστερα θέτουμε τις ακόλουθες επιλογές για το The Things Uno:

- **End device brand:** The Things Product
- **Model:** The Things Uno
- **Hardware Ver.:** 1.0
- **Firmware Ver.:** quickstart
- **Profile (Region):** EU\_863\_870
- **Frequency plan:** Europe 863-870MHz (SF9 for RX2 - recommended)

## Register end device

Does your end device have a LoRaWAN® Device Identification QR Code? Scan it to speed up onboarding.

 Scan end device QR code

 Device registration help

### End device type

Input method 

Select the end device in the LoRaWAN Device Repository

Enter end device specifics manually

End device brand  *	Model  *	Hardware Ver.  *	Firmware Ver.  *	Profile (Region)  *
The Things Prod... 	The Things Uno 	1.0 	quickstart 	EU_863_870 

### The Things Uno

 LoRaWAN Specification 1.0.2, RP001 Regional Parameters 1.0.2 revision B, Over the air activation (OTAA), Class A

The Things Uno is based on the Arduino Leonardo with an added Microchip LoRaWAN® module. It is fully compatible with the Arduino IDE and existing shields.

Frequency plan  \*

Europe 863-870 MHz (SF9 for RX2 - recommended) 

Εικόνα 5.22: Επιλογή The Things Uno μοντέλου από Device registry του The Things Stack.

3. Έπειτα, ακριβώς από κάτω, μας ανοίγονται ορισμένα πεδία προς συμπλήρωση που αποτελούν τις πληροφορίες παροχής (provisioning information):

- **JoinEUI (Πρώην AppEUI):** την τιμή AppEUI που βρήκαμε τρέχοντας το sketch Deviceinfo (Εικόνα 5.18). Την ίδια τιμή χρησιμοποιούμε και στο Arduino sketch μας.
- **DevEUI:** την τιμή DevEUI που βρήκαμε τρέχοντας το sketch Deviceinfo (Εικόνα 5.18).
- **AppKey:** πατάμε το κουμπί Generate για να δημιουργήσουμε κλειδί κρυπτογράφησης για το OTAA. Αποθηκεύουμε το κλειδί που παράγεται προκειμένου να το χρησιμοποιήσουμε στο Arduino sketch μας.

- **End Device ID:** βάζουμε ένα μοναδικό αναγνωριστικό για την τερματική συσκευή μας.

Provisioning information

JoinEUI ⓘ \*

Reset

This end device can be registered on the network

DevEUI ⓘ \*

AppKey ⓘ \*

Generate

End device ID ⓘ \*

After registration

View registered end device

Register another end device of this type

**Register end device**

Εικόνα 5.23: Συμπλήρωση Provisioning information για το νέο end device.

- Τέλος, πατάμε *Register end device* και ολοκληρώνουμε την διαδικασία. Πλέον εμφανίζεται η συσκευή μας στο μενού *End devices* της εφαρμογής:

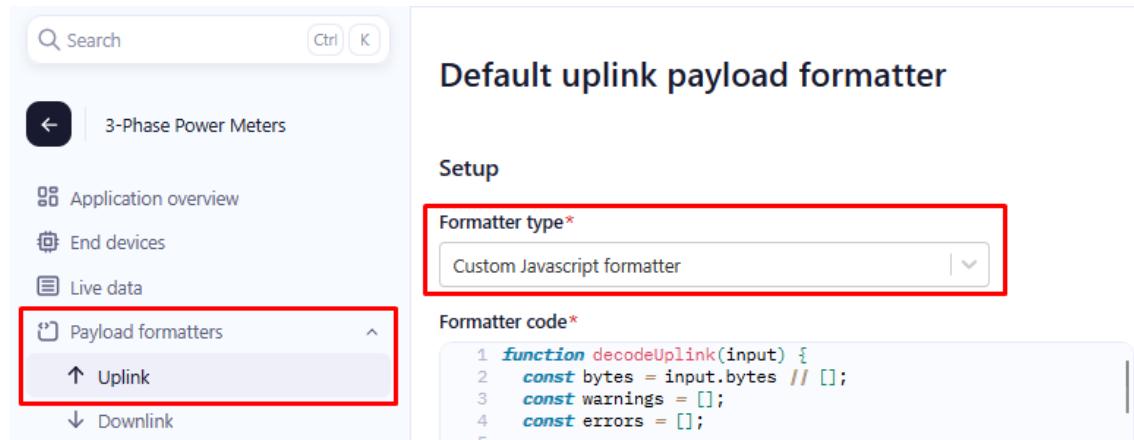
NAME AND ID :	DEVEUI	JOINERI	LAST ACTIVITY :
eui-0004a30b001bdb64	00 04 A3 0B 00 1B DB 64	00 00 00 00 00 00 00 00	Never *

Εικόνα 5.24: Η καταχωρημένη τερματική συσκευή στο μενού της εφαρμογής.

#### 5.2.4 Κωδικοποίηση Payload Formatter στο TTS

Για να μετατραπούν τα ακατέργαστα bytes του uplink σε δομημένα μεγέθη (τάση, ρεύμα, ισχύς κ.λπ.), υλοποιείται ένας Payload Formatter σε JavaScript μέσα στο Console του The Things Stack. Ο formatter είναι πλήρως συγχρονισμένος με τον encoder του Arduino sketch μας (14 bytes/φάση: uint16 κλιμακωμένα σε big-endian για όλα τα μεγέθη εκτός της ενέργειας που αποστέλλεται ως float32 big-endian). Υποστηρίζει από 1 έως 3 φάσεις, κάνει ανάγνωση με σταθερά offsets, μετατρέπει τις τιμές στις αρχικές μονάδες (π.χ. V, A, Hz) και χειρίζεται sentinel τιμές (0xFFFF/0xFFFFFFFF) επιστρέφοντας null όταν κάποια μέτρηση είναι άκυρη (NaN/Inf/εκτός εύρους). Επιπλέον, εκδίδει προειδοποιήσεις για μη αναμενόμενα μήκη payload ή πεδία που κατέληξαν null, διευκολύνοντας τη διάγνωση. Το αποτέλεσμα είναι σε JSON μορφή και είναι έτοιμο να οδηγηθεί απευθείας σε webhooks, όπως θα δουμε αργότερα.

Από το μενού αριστερά της εφαρμογής διαλέγουμε *Payload Formatters > Uplink*. Εδώ ορίζουμε τον βασικό (default) payload formatter που θα χρησιμοποιούν όλες οι εγγεγραμμένες τερματικές συσκευές στην εφαρμογή (μπορούμε να θέσουμε και αποκλειστικό payload formatter για κάθε συσκευή). Επιλέγουμε αρχικά **Formatter type:** *Custom Javascript formatter* και συμπληρώνουμε τον κώδικα που ακολουθεί παρακάτω. Στο τέλος πατάμε **Save changes**.



Εικόνα 5.25: Κωδικοποίηση του Default uplink payload formatter.

### Κώδικας payload formatter

```

1  function decodeUplink(input) {
2    const bytes = input.bytes || [];
3    const warnings = [];
4    const errors = [];

5    // ---- Helpers ----
6    const READS_PER_PHASE = 14;

7    function readBE16(b, i) {
8      return (b[i] << 8) | b[i + 1];
9    }

10   function readBEu32(b, i) {
11     return ((b[i] << 24) >>> 0) | (b[i + 1] << 16) | (b[i + 2] << 8) | (b[i + 3] << 0);
12   }

13   function readBEfloat32(b, i) {
14     // Build a big-endian float32 from 4 bytes
15     const u8 = new Uint8Array(4);
16     u8[0] = b[i];
17     u8[1] = b[i + 1];
18     u8[2] = b[i + 2];
19     u8[3] = b[i + 3];
20     const dv = new DataView(u8.buffer);
21     return dv.getFloat32(0, false); // big-endian
22   }
23
24 }
```

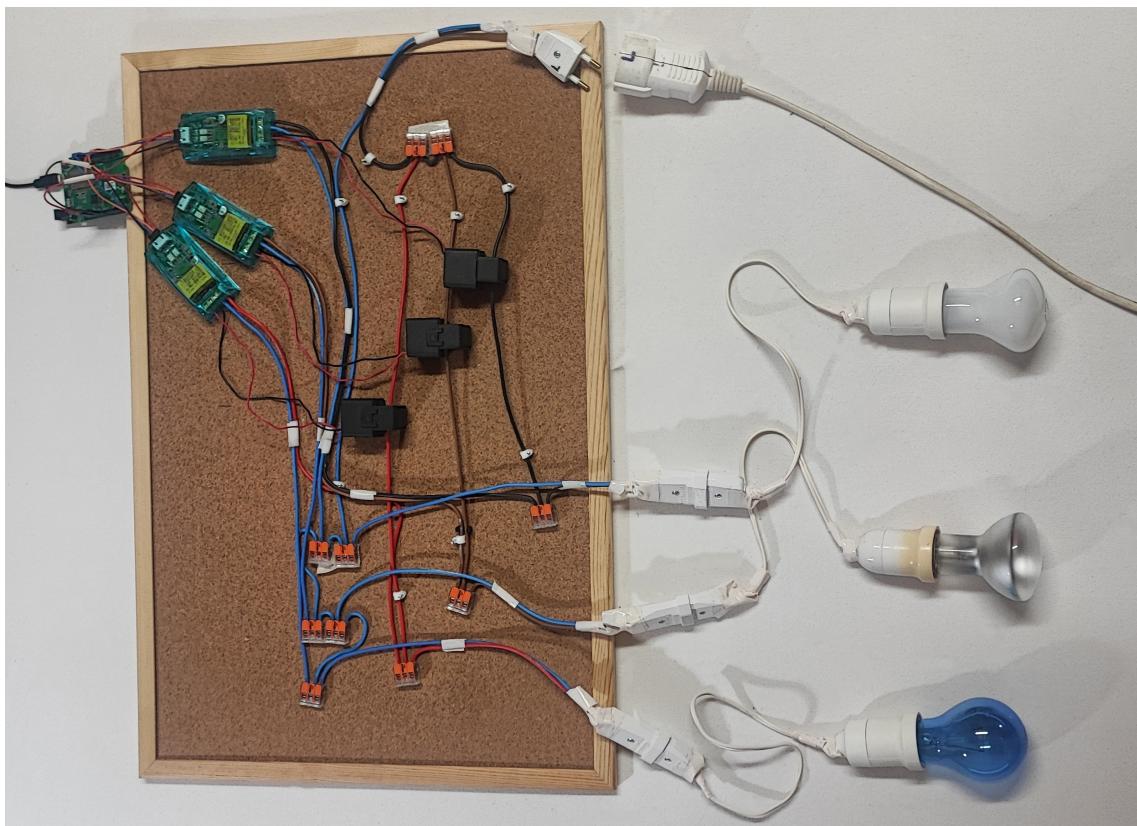
```

28   function decodePhase(b, base) {
29     // Offsets per Arduino encoder
30     const vRaw = readBE16(b, base + 0); // V * 10
31     const iRaw = readBE16(b, base + 2); // A * 100
32     const pRaw = readBE16(b, base + 4); // W * 10
33     const eU32 = readBEu32(b, base + 6); // float32 (raw, BE) sentinel check
34     const fRaw = readBE16(b, base + 10); // Hz * 10
35     const pfRaw = readBE16(b, base + 12); // * 100
36     const INVALID16 = 0xFFFF >>> 0;
37     const INVALID32 = 0xFFFFFFFF >>> 0;
38
39   return {
40     voltage: vRaw === INVALID16 ? null : vRaw / 10, // V
41     current: iRaw === INVALID16 ? null : iRaw / 100, // A
42     power: pRaw === INVALID16 ? null : pRaw / 10, // W
43     energy: eU32 === INVALID32 ? null : readBEfloat32(b, base + 6), // kWh
44     (raw float)
45     frequency: fRaw === INVALID16 ? null : fRaw / 10, // Hz
46     powerFactor: pfRaw === INVALID16 ? null : pfRaw / 100 // unitless
47   };
48 }
49
50 // ---- Decode ----
51 const phases = Math.min(3, Math.floor(bytes.length / READS_PER_PHASE));
52 if (phases === 0) {
53   errors.push("Not enough bytes: expected at least 14.");
54   return { data: {}, warnings, errors };
55 }
56
57 const decoded = {};
58 for (let i = 0; i < phases; i++) {
59   const base = i * READS_PER_PHASE;
60   const sensor = decodePhase(bytes, base);
61   decoded["sensor" + (i + 1)] = sensor;
62 }
63
64 if (bytes.length !== phases * READS_PER_PHASE) {
65   warnings.push(
66     `Unexpected length: got ${bytes.length} bytes; decoded ${phases} phase(s) =
67     ${phases * READS_PER_PHASE} bytes.`);
68 }
69
70 // Warn if any sentinel (null) is present
71 const anyNull =
72   Object.values(decoded).some(s =>
73     Object.values(s).some(v => v === null)
74   );
75 if (anyNull) {
76   warnings.push("One or more fields are invalid/null (sentinel received.).");
77 }
78
79 return { data: decoded, warnings, errors };

```

### 5.3 Πειραματική δοκιμή λειτουργείας

Έχοντας πλέον ρυθμίσει το LoRaWAN gateway και προετοιμάσει το The Things Stack για τη σύνδεση και επικοινωνία με τον τριφασικό μετρητή, μπορούμε να ξεκινήσουμε την δοκιμή του συστήματος. Για αρχή κατασκευάζουμε ένα απλό κύκλωμα με τρεις λαμπτήρες ώστε να προσομοιώσουμε μία τριφασική κατανάλωση (Εικόνα 5.24). Συγκεκριμένα, από ένα ρευματολήπτη (pulg) απομονώνουμε τη φάση **L** (Live) και τον ουδέτερο **N** (Neutral). Στη συνέχεια διακλαδώνουμε την φάση σε τρεις κλάδους (με συνδέσμους WAGO), όπου κάθε κλάδος αντιστοιχεί σε μία «υπο-φάση» της προσομοίωσης και στην κάθε μία συνδέουμε τον ακροδέκτη **L** του αντίστοιχου PZEM και εναν λαμπτήρα. Όμοια, συνδέουμε τον ουδέτερο του ρευματολήπτη με τον ακροδέκτη **N** όλων των PZEM και με όλους τους λαμπτήρες. Τέλος, ο μετασχηματιστής ρεύματος του PZEM (split CT) συνδέται στην αντίστοιχη «υπο-φάση», αμέσως μετά το σημείο που έγινε η διακλάδωση. Με αυτή τη συνδεσμολογία καταγράφονται τόσο **οι καταναλώσεις των λαμπτήρων** όσο και **οι ίδιες οι απώλειες/καταναλώσεις των PZEM**, παρέχοντας σαφή εικόνα συμπεριφοράς του τριφασικού μετρητή σε συνθήκες δοκιμής.



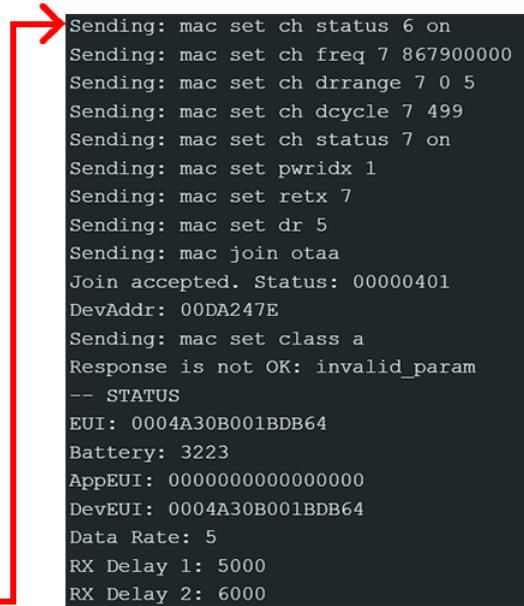
Εικόνα 5.26: Δοκιμαστικό κύκλωμα τριφασικού μετρητή με τρεις λαμπτήρες ως φορτία (φάσεις L1-L3). Το καλώδιο φάσης του ρευματολήπτη (μάυρο) χωρίζεται σε 3 καλώδια (κόκκινο, καφέ, μαύρο), τα οποία συνδέονται με ένα PZEM και ένα λαμπτήρα, αντιπροσωπεύοντας μία «υπο-φάση» το καθένα. Το καλώδιο ουδετέρου του ρευματολήπτη (μπλε) συνδέεται σε όλα τα PZEM και τους λαμπτήρες.

### Συνδεσμολογία δοκιμής (PZEM-λαμπτήρας):

- **PZEM 1** (κλάδος φάσης, κόκκινο καλώδιο): OSRAM Bellalux **75Ω/230V** (λευκός λαμπτήρας πυρακτώσεως). Ακροδέκτης L του PZEM στον αντίστοιχο κλάδο φάσης, ακροδέκτης N στον κοινό ουδέτερο. Ο split CT τυλίγει τον ίδιο αγωγό L με φορά «source → load».
- **PZEM 2** (κλάδος φάσης, καφέ καλώδιο): Tungsram R63 **60Ω/240V** (ανακλαστήρας spot). Σύνδεση L/N όπως ανωτέρω, CT στον καφέ αγωγό προς το φορτίο.
- **PZEM 3** (κλάδος φάσης, μαύρο καλώδιο): Philips **60Ω/230V** (μπλε λαμπτήρας πυρακτώσεως). Σύνδεση L/N όπως ανωτέρω, CT στον μαύρο αγωγό προς το φορτίο.

**Σημείωση καλωδίωσης:** Και τα τρία PZEM-004T v3.0 τροφοδοτούνται/μετρούν από τους ακροδέκτες L/N (ο ουδέτερος είναι κοινός), ενώ ο μετασχηματιστής ρεύματος κάθε PZEM τοποθετείται μόνο στον αντίστοιχο αγωγό φάσης. Η φορά του CT πρέπει να είναι σταθερά από την πλευρά τροφοδοσίας προς το φορτίο, ώστε το ρεύμα να μετριέται με σωστό πρόστημα. Τα ονομαστικά 230/240V των λαμπτήρων εξηγούν μικρές διαφορές στην αποδιδόμενη ισχύ, δεδομένου ότι για ωμικά φορτία ισχύει κατά προσέγγιση  $P \propto V^2$ .

Στη συνέχεια κάνουμε upload (βλ. Εικόνα 5.17) το Arduino sketch της Υπο-υποενότητας 5.2.2.2 στο The Things Uno και συνδέουμε τον ρευματολήπτη στην πρήζα (230V), με προσοχή στις πολικότητες. Έπειτα κοιτάμε το Serial monitor και βλέπουμε:



```

Sending: mac set rx2 3 869525000
Sending: mac set ch drrange 1 0 6
Sending: mac set ch dcycle 0 299
Sending: mac set ch dcycle 1 299
Sending: mac set ch dcycle 2 299
Sending: mac set ch freq 3 867100000
Sending: mac set ch drrange 3 0 5
Sending: mac set ch dcycle 3 499
Sending: mac set ch status 3 on
Sending: mac set ch freq 4 867300000
Sending: mac set ch drrange 4 0 5
Sending: mac set ch dcycle 4 499
Sending: mac set ch status 4 on
Sending: mac set ch freq 5 867500000
Sending: mac set ch drrange 5 0 5
Sending: mac set ch dcycle 5 499
Sending: mac set ch status 5 on
Sending: mac set ch freq 6 867700000
Sending: mac set ch drrange 6 0 5
Sending: mac set ch dcycle 6 499
Sending: mac set ch status 6 on

```

```

Sending: mac set ch status 6 on
Sending: mac set ch freq 7 867900000
Sending: mac set ch drrange 7 0 5
Sending: mac set ch dcycle 7 499
Sending: mac set ch status 7 on
Sending: mac set pwridx 1
Sending: mac set retx 7
Sending: mac set dr 5
Sending: mac join otaa
Join accepted. Status: 00000401
DevAddr: 00DA247E
Sending: mac set class a
Response is not OK: invalid_param
-- STATUS
EUI: 0004A30B001BDB64
Battery: 3223
AppEUI: 0000000000000000
DevEUI: 0004A30B001BDB64
Data Rate: 5
RX Delay 1: 5000
RX Delay 2: 6000

```

Εικόνα 5.27: Ροή εντολών MAC και επιτυχές OTAA join της συσκευής στο EU868 (ρυθμίσεις καναλιών/ρυθμών, DevAddr, DR5, αποτυχημένη αλλαγή κλάσης ως αναμενόμενο).

Στο στιγμιότυπο του Serial Monitor παραπάνω φαίνεται η πλήρης ακολουθία ρύθμισης και ένταξης του The Things Uno στο LoRaWAN δίκτυο. Πρώτα στέλνονται **εντολές MAC** που ορίζουν τις παραμέτρους για **ΕΥ868: rx2 3 869525000** (κανάλι RX2 στα 869.525MHz, DR3), εύρη δεδομένων (drrange) και dcycle για τα κανάλια 1-7, καθώς και οι συχνότητες 867.1-867.9MHz. Κατόπιν τίθενται **pwridx 1** (δείκτης ισχύος εκπομπής), **retx 7** (μέγιστες

επαναμεταδόσεις) και **dr 5** (ρυθμός δεδομένων **DR5**, δηλ. SF7BW125 στο EU868). Ακολουθεί **OTAA join**: «Join accepted. Status: 00000401» και εκχώρηση **DevAddr** (00DA247E). Η εντολή «**mac set class a**» επιστρέφει **invalid\_param** επειδή τα RN2483/RN2903 είναι εγγενώς *Class A* και επομένως δεν απαιτούν/δέχονται αλλαγή κλάσης μέσω αυτής της εντολής. Το μπλοκ «– **STATUS**» εμφανίζει EUI/DevEUI/AppEUI, τον ενεργό **Data Rate: 5**, και τις καθυστερήσεις **RX1/RX2** (5000/6000ms) που θα ισχύουν για τα downlinks. Μετά την επιτυχή ένταξη, ο κώδικας στέλνει περιοδικά uplink με το πακετάρισμα των μετρήσεων από τα τρία PZEM:

```
PZEM 1 Readings:
Voltage: 232.50 V
Current: 0.335 A
Power: 77.20 W
Energy: 0.0020 kWh
Frequency: 50.00 Hz
Power Factor: 0.99

PZEM 2 Readings:
Voltage: 232.40 V
Current: 0.235 A
Power: 53.30 W
Energy: 0.0010 kWh
Frequency: 50.00 Hz
Power Factor: 0.98

PZEM 3 Readings:
Voltage: 232.60 V
Current: 0.257 A
Power: 58.70 W
Energy: 0.0020 kWh
Frequency: 49.90 Hz
Power Factor: 0.98

-- Sending data to TTN
Sending: mac tx uncnf 1 0915002203043B03126F01F400630914001802153A83126F01F400620916001A024B3B03126F01F30062
```

Εικόνα 5.28: Ένδειξη Serial Monitor με μετρήσεις από 3 PZEM-004T και αποστολή uplink (**mac tx uncnf**) στο TTN.

Στην Εικόνα 5.28 φαίνονται οι πλήρεις μετρήσεις ανά φάση από τους τρεις αισθητήρες PZEM (τάση, ρεύμα, ισχύς, ενέργεια, συχνότητα, συντελεστής ισχύος) πριν από κάθε αποστολή. Για παράδειγμα, ο **PZEM 1** μετρά 232.50V, 0.335A και 77.20W, ενώ ο **PZEM 2** και **PZEM 3** δίνουν 53.30W και 58.70W αντίστοιχα. Αμέσως μετά, ο κώδικας πακετάρει τις τιμές (σύνολο **42 bytes**: 14 bytes/phase, big-endian) και στέλνει το **uplink** με την εντολή **mac tx uncnf 1 <hex payload>**, η οποία εμφανίζεται στο τέλος ως δεκαεξαδική ακολουθία.

Οι ενδείξεις της Εικόνας 5.28 είναι πλήρως συνεπείς με τη φυσική των ωμικών φορτίων (incandescent lamps): ο συντελεστής ισχύος παραμένει ( $\approx 1$ ) ( $PF=0.98-0.99$ ), η συχνότητα σταθερή κοντά στα (50Hz), ενώ η ενεργός ισχύς προσεγγίζει άριστα τον θεωρητικό υπολογισμό ( $P \approx V \cdot I$ ) για κάθε κλάδο. Για τον OSRAM 75 W/230 V (PZEM 1) η μετρούμενη τιμή (77.2W) εξηγείται από την ελαφρώς αυξημένη τάση τροφοδοσίας (~ 232.5V) και τις ονομαστικές ανοχές του λαμπτήρα. Για τον Tungsram R63 60W/240V (PZEM 2) η χαμηλότερη τάση (~ 232.4V) οδηγεί σε ισχύ μικρότερη της ονομαστικής, όπως αναμένεται από τον κανόνα ( $P \propto V^2$ ) (θεωρητικά (~ 56W), μετρημένα (53.3W)). Ο Philips 60 W/230 V (PZEM 3) αποδίδει (58.7W), εντός φυσιολογικών ανοχών και με μικρή επίδραση της θερμοκρασίας του νήματος. Τέλος, οι σωρευτικές τιμές energy (0.001-0.002kWh) (δηλ. (1-2Wh) αντιστοιχούν

σε λειτουργία λίγων λεπτών, επιβεβαιώνοντας ότι τόσο η κωδικοποίηση/αποστολή του uplink όσο και η μέτρηση των τριών PZEM λειτουργούν ορθά και επαναλήψιμα στο παρόν στήσιμο.

### Παραλαβή και αποκωδικοποίηση στο The Things Stack

Με το που εκπέμψει το TTU το uplink, λαμβάνεται από το Basic Station, το οποίο το προ-ωθεί (forward) μέσω LNS προς τον Gateway Server του The Things Stack. Εκεί, το μήνυμα ελέγχεται, αποκρυπτογραφείται από τον Network Server και αποκωδικοποιείται σε φυσικές μονάδες (decoded\_payload). Η όλη ροή αποτυπώνεται ζωντανά στην καρτέλα Live data της συσκευής (βλ. Εικόνα 5.29): πρώτα η αποδοχή ένταξης (Accept join-request/Successfully processed join-accept), στη συνέχεια η προώθηση και επιτυχής επεξεργασία των uplinks (Forward uplink data message/Successfully processed data up message), μαζί με προεπι-σκόπηση του FRMPayload (δεκαεξαδικό) και του αποτελέσματος της αποκωδικοποίησης.

TIME	TYPE	DATA PREVIEW
18:24:46	Update end device	[ "activated_at" ]
↑ 19:26:24	Forward uplink data message	DevAddr: 00 DA 24 7E Payload: { sensor1: ..., sensor2: ..., sensor3: ... } 09 15 00 22 03 04 3B 03 12 6F 01 F4 00 6
↑ 19:26:23	Successfully processed dat...	DevAddr: 00 DA 24 7E
↑ 19:26:20	Forward join-accept message	DevAddr: 00 DA 24 7E JoinEUI: 00 00 00 00 00 00 00 DevEUI: 00 04 A3 08 00 1B DB 64
↑ 19:26:18	Successfully processed jo...	DevAddr: 01 9E 17 01 JoinEUI: 00 00 00 00 00 00 DevEUI: 00 04 A3 08 00 1B DB 64
↑ 19:26:18	Accept join-request	DevAddr: 00 DA 24 7E JoinEUI: 00 00 00 00 00 00 DevEUI: 00 04 A3 08 00 1B DB 64

Εικόνα 5.29: Πίνακας Live data στο TTS: χρονολόγιο γεγονότων για join-accept και εισερχόμενα uplinks. Διακρίνονται τα πεδία DevAddr, η επισήμανση Payload με δεκαεξαδικά δεδομένα, καθώς και η προεπισκόπηση decoded\_payload όταν είναι ενεργός ο formatter.

Στην Εικόνα 5.29 φαίνεται αναλυτικά το χρονικό των μηνυμάτων. Δεξιά, στη στήλη DATA PREVIEW, διακρίνεται το DevAddr που αποδόθηκε κατά το OTAA, ενώ για κάθε uplink εμφανίζεται το ωφέλιμο FRMPayload σε μορφή hex. Αν η λειτουργία Verbose stream είναι ενεργοποιημένη, προβάλλονται επιπλέον πληροφορίες (π.χ. χρόνοι, τύποι γεγονότων, κ.ά.), χρήσιμες για έλεγχο και αποσφαλμάτωση. Τα κουμπιά Pause/Clear/Export as JSON διευκολύνουν την παύση της ροής, την εκκαθάριση της λίστας και την εξαγωγή των γεγονότων αντίστοιχα.

Ακολούθως, ο Uplink Payload Formatter μας εφαρμόζει τη σύμβαση κωδικοποίησης που ορίσαμε στο TTU (κλίμακες τιμών, big-endian πακετάρισμα, sentinel για NaN/Inf ενNεγ) και παράγει ένα δομημένο αντικείμενο JSON με τις τρεις φάσεις. Στην Εικόνα 5.30 φαίνονται οι μετρήσεις ανά αισθητήρα PZEM (sensor1-sensor3) σε φυσικές μονάδες, ακριβώς όπως θα τις καταναλώσει λάθει η διαδικτυακή εφαρμογή που θα αναπτύξουμε στη συνέχεια (Κεφάλαιο 3), μέσω HTTP webhook.

Παρατηρούμε ότι τα πεδία του decoded\_payload (τάση γύρω στα 232-233 V, ρεύμα 0.24-0.34 A, ισχύς 49-77 W, συχνότητα ~ 50 Hz, PF 0.98-0.99, ενέργεια 1-2 Wh για λίγα λεπτά

```

34   "uplink_message": [
35     "session_key_id": "AZor2zdXHV50eGuJmsNzQ==",
36     "f_port": 1,
37     "frm_payload": "CRUAIgME0wMSbwH0AGMJFAAYAhU6gxJvAfQAYgkWABoCSzsI",
38     "decoded_payload": [
39       "sensor1": {
40         "current": 0.34,
41         "energy": 0.002000000949949026,
42         "frequency": 50,
43         "power": 77.2,
44         "powerFactor": 0.99,
45         "voltage": 232.5
46       },
47       "sensor2": {
48         "current": 0.24,
49         "energy": 0.001000000474974513,
50         "frequency": 50,
51         "power": 53.3,
52         "powerFactor": 0.98,
53         "voltage": 232.4
54       },
55       "sensor3": {
56         "current": 0.26,
57         "energy": 0.002000000949949026,
58         "frequency": 49.9,
59         "power": 58.7,
60         "powerFactor": 0.98,
61         "voltage": 232.6
62     }
63   ],

```

Εικόνα 5.30: *decoded\_payload* από τον *formatter*: ανά *sensor* προβάλλονται *voltage*, *current*, *power*, *energy*, *frequency*, *powerFactor*. Οι τιμές προκύπτουν από τις κλίμακες κωδικοποίησης του *uplink* και είναι έτοιμες για αποδήκευση/οπτικοποίηση χωρίς περαιτέρω μετασχηματισμούς.

λειτουργίας) ταυτίζονται με τις ενδείξεις του Serial Monitor και επιβεβαιώνουν ότι η διαδρομή Device → Basic Station → TTS είναι ορθή. Το γεγονός ότι η αποκωδικοποίηση γίνεται εντός του TTS σημαίνει ότι η δοωνστρεαμ κατανάλωση δεδομένων είναι απλούστερη: είτε μέσω webhooks (JSON POST στο backend) είτε μέσω MQTT οι ίδιες τιμές διατίθενται ήδη σε φυσικές μονάδες, μειώνοντας την πολυπλοκότητα στην εφαρμογή. Επιπλέον, η προβολή στο Live data επιτρέπει άμεσο εντοπισμό σφαλμάτων, όπως ασυνεπές FPort, μηδενικό payload, λανθασμένη κλίμακα ή αποτυχία του formatter, που είναι χρήσιμο να γνωρίζουμε πριν προχωρήσουμε στην ολοκλήρωση με το πληροφοριακό μας σύστημα.

## Ανάπτυξη Web Εφαρμογής για την οπτικοποίηση των δεδομένων

---

Στο κεφάλαιο αυτό παρουσιάζεται η web εφαρμογή που αναπτύχθηκε για την εποπτεία του τριφασικού συστήματος μέτρησης, από το επίπεδο του διακομιστή (backend) μέχρι το περιβάλλον χρήστη (frontend). Η εφαρμογή αναλαμβάνει να συλλέγει τα uplink μηνύματα που δρομολογούνται από το The Things Stack προς ένα HTTP webhook, να τα αποθηκεύει σε σχεσιακή βάση δεδομένων και να τα παρουσιάζει σε γραφική μορφή σε πίνακες και διαγράμματα. Η υλοποίηση χωρίζεται σε δύο κύρια υποσυστήματα: μία Spring Boot εφαρμογή backend σε Java (με PostgreSQL και ασφάλεια βασισμένη σε JSON Web Tokens) και μία React εφαρμογή frontend σε TypeScript που «τρέχει» στον φυλλομετρητή και επικοινωνεί με το REST API του backend μέσω HTTP JSON κλήσεων.

Στις επόμενες ενότητες περιγράφεται λεπτομερώς η αρχιτεκτονική της εφαρμογής, η δομή των βασικών κλάσεων και endpoints, η ροή των δεδομένων από το The Things Stack μέχρι την βάση δεδομένων και το UI, καθώς και οι επιλογές ασφάλειας και ανάπτυξης στο LoRaWAN gateway (το Raspberry Pi της εγκατάστασης).

### 6.1 Γενική αρχιτεκτονική εφαρμογής

Η web εφαρμογή έχει σχεδιαστεί με την εξής τριεπίπεδη λογική:

- Επίπεδο δικτύου LoRaWAN:** Οι τριφασικοί μετρητές στέλνουν περιοδικά uplinks μέσω του LoRaWAN gateway στο The Things Stack, όπου γίνεται η αποκρυπτογράφηση του LoRaWAN πακέτου, η εφαρμογή του payload formatter και η παραγωγή JSON δομής με τις φυσικές μετρήσεις (τάση, ρεύμα, ισχύς, ενέργεια, συχνότητα, συντελεστής ισχύος ανά φάση), όπως περιγράφηκε στα προηγούμενα κεφάλαια.
- Επίπεδο backend (Spring Boot REST API):** Ένα HTTP webhook που εκτίθεται από την εφαρμογή backend (σε συγκεκριμένη URL) δέχεται τα JSON μηνύματα (uplink) από το TTS. Τα μηνύματα αυτά χαρτογραφούνται σε αντικείμενα DTO (Data Transfer Object), μετατρέπονται σε οντότητες JPA και αποθηκεύονται στη βάση δεδομένων PostgreSQL. Παράλληλα, το ίδιο REST API παρέχει προστατευμένα endpoints για ανάκτηση ιστορικών δεδομένων (π.χ. ανά μετρητή, ανά χρονικό διάστημα) από την React εφαρμογή.

- **Επίπεδο frontend (React SPA):** Το frontend υλοποιείται ως Single Page Application (SPA) με React και TypeScript. Ο χρήστης εισέρχεται στο σύστημα με όνομα χρήστη και κωδικό (μία διαχειριστική εγγραφή), λαμβάνει JWT token και στη συνέχεια μπορεί να βλέπει σε πραγματικό χρόνο (ή με περιοδικά refresh) τις μετρήσεις των τριφασικών μετρητών σε διαγράμματα γραμμής, ράθδων ή πίτας, καθώς και σε πίνακες με χρονοσήμανση.

Η συνολική αρχιτεκτονική είναι πλήρως ανεξάρτητη από το LoRaWAN υπόστρωμα: το backend βλέπει τα δεδομένα μόνο ως HTTP JSON μηνύματα από το TTS, ενώ το frontend θεωρεί το backend ως μια τυπική REST υπηρεσία.

## 6.2 Backend: Spring Boot REST API και ασφάλεια

Το backend υλοποιείται ως Spring Boot εφαρμογή σε Java 17, με Maven για τη διαχείριση των εξαρτήσεων, Spring Web για το REST API, Spring Data JPA για την πρόσθαση στη βάση δεδομένων PostgreSQL και Liquibase για τον έλεγχο των αλλαγών στο σχήμα της βάσης. Για την ασφάλεια χρησιμοποιείται αυθεντικοποίηση με JWT filter που παρεμβάλλεται στην Spring Security αλυσίδα και ελέγχει κάθε εισερχόμενο αίτημα προς τα προστατευμένα endpoints.

### 6.2.1 Δομή πακέτων, βασικών κλάσεων και βάσης δεδομένων

Η εφαρμογή ακολουθεί μια κλασική στρωματοποιημένη δομή:

- **controller πακέτα:** Περιλαμβάνουν τις REST controllers για τα δεδομένα μετρήσεων (π.χ. ανάγνωση χρονοσειρών για έναν μετρητή) και για την αυθεντικοποίηση (login και έκδοση JWT).
- **service πακέτα:** Υλοποιούν την επιχειρησιακή λογική (business logic), όπως την επεξεργασία των uplink μηνυμάτων από το TTS, τον υπολογισμό ή την φιλτράρισμά τους ανά ημερομηνία, καθώς και την διαχείριση χρηστών.
- **repository πακέτα:** Ορίζουν τις διεπαφές JPA repositories που διαχειρίζονται οντότητες όπως SensorData, Meter και User, χαρτογραφημένες σε πίνακες της PostgreSQL.
- **security πακέτο:** Περιέχει τις κλάσεις SecurityConfig, JWT βοηθητικές συναρτήσεις (π.χ. παραγωγή και επαλήθευση tokens) και το filter που ελέγχει την επικεφαλίδα "Authorization: Bearer <token>" σε κάθε αίτημα.

Το σχήμα της βάσης δεδομένων ελέγχεται από αρχεία Liquibase changelog, στα οποία περιγράφονται οι πίνακες users (μοντέλο χρήστη με username και password), οι πίνακες μετρητών και δεδομένων, καθώς και τυχόν περιορισμοί, indexes και σχέσεις (π.χ. ένα πλήθος εγγραφών δεδομένων ανά μετρητή). Επιπλέον προστίθενται με αυτόν το τρόπο τα δεδόμένα (πιστοιτικά) για τον admin χρήστη και οι βασικές πληγροφορίες για τους 2 μετρητές μας. Η πίνακες της βάσης έχουν την εξής μορφή:

Όνομα πεδίου	Τύπος	Περιγραφή
id	bigint (auto increment)	Μοναδικό αναγνωριστικό εγγραφής μετρητή (πρωτεύον κλειδί).
meter_id	varchar(50)	Λογικό αναγνωριστικό του μετρητή, όπως έχει δηλωθεί στο The Things Stack (device_id).
location	varchar(100)	Περιγραφή θέσης εγκατάστασης του μετρητή (π.χ. «MV Substation», «Lab bench»).
created_at	timestamp	Ημερομηνία και ώρα δημιουργίας της εγγραφής (προεπιλογή now()).

Πίνακας 6.1: Πεδία του πίνακα meter.

Όνομα πεδίου	Τύπος	Περιγραφή
id	bigint (auto increment)	Μοναδικό αναγνωριστικό εγγραφής μέτρησης (πρωτεύον κλειδί).
meter_id	bigint	Ξένο κλειδί προς τον πίνακα meter (συσχέτιση μέτρησης με συγκεκριμένο μετρητή).
sensor_id	varchar(50)	Αναγνωριστικό φάσης/καναλιού (sensor1, sensor2, sensor3).
current	double precision	Ρεύμα φάσης σε A.
energy	double precision	Ενέργεια φάσης σε kWh.
frequency	double precision	Συχνότητα δικτύου σε Hz.
power	double precision	Ενεργός ισχύς φάσης σε W.
power_factor	double precision	Συντελεστής ισχύος (power factor) φάσης (αδιάστατο μέγεθος).
voltage	double precision	Τάση φάσης σε V.
timestamp	timestamp	Χρονοσήμανση λήψης/αποθήκευσης της συγκεκριμένης μέτρησης.

Πίνακας 6.2: Πεδία του πίνακα sensor\_data.

Όνομα πεδίου	Τύπος	Περιγραφή
id	bigint (auto increment)	Μοναδικό αναγνωριστικό χρήστη (πρωτεύον κλειδί).
username	varchar(255)	Όνομα χρήστη (login) του διαχειριστή της εφαρμογής.
password	varchar(255)	Κωδικός πρόσθασης του χρήστη, αποθηκευμένος σε μορφή hash (bcrypt).

Πίνακας 6.3: Πεδία του πίνακα users.

## 6.2.2 Διαχείριση εισερχόμενων uplinks μέσω webhook

Κεντρικό ρόλο στην πλατφόρμα παίζει το webhook endpoint στο οποίο το The Things Stack στέλνει τα uplink μηνύματα. Η διαδικασία έχει ως εξής:

- Στο The Things Stack Console έχουμε δημιουργήσει ένα Application όπου έχουν καταχωρηθεί οι τριφασικοί μετρητές ως end devices. Για την εφαρμογή αυτή ενεργοποιείται ένας HTTP webhook που στο πεδίο Base URL δείχνει προς την URL του backend (<http://192.168.0.100:8080/lorawan-data>). Προκειμένου να δημιουργήσουμε ένα νέο webhook ακολουθούμε την εξής διαδικασία:

Στην κονσόλα του TTS ανοίγουμε την εφαρμογή μας → Integrations → Webhooks → + Add webhook. Διαλέγουμε Custom Webhook, ορίζουμε Webhook ID και Base URL (τελικό endpoint του backend). Στο πεδίο Enable event types επιλέγουμε Uplink message και πατάμε Add webhook για να ολοκληρώσουμε την διαδικασία.

ID	BASE URL	TEMPLATE ID	CREATED AT
power-monitoring-app-webhook	<a href="http://192.168.0.100:8080/lorawan-data">http://192.168.0.100:8080/lorawan-data</a>	None	Feb 22, 2025

Εικόνα 6.1: Στην εφαρμογή 3-Phase Power Meters, φαίνεται το ενεργό webhook με ID power-monitoring-app-webhook και Base URL <http://192.168.0.100:8080/lorawan-data>.

- Για κάθε uplink μήνυμα, το TTS στέλνει ένα HTTP POST προς το webhook με σώμα JSON. Στο body περιλαμβάνονται μεταδεδομένα (end\_device\_ids, received\_at, rx\_metadata κ.λπ.) και το decoded\_payload όπως το επιστρέφει το payload formatter (με πεδία sensor1, sensor2, sensor3 κ.ά.).
- Στον backend, ένας controller δέχεται το POST, το χαρτογραφεί σε DTO κλάσεις και εξάγει τα απαραίτητα πεδία:
  - το αναγνωριστικό του μετρητή (device\_id),
  - τις τιμές ανά φάση από τα sensor1, sensor2, sensor3,
  - τη χρονοσήμανση received\_at.
- Οι τιμές αυτές μετατρέπονται σε οντότητες SensorData, με πεδία όπως sensorId, voltage, current, power, energy, frequency, powerFactor, timestamp, και αποθηκεύονται στη βάση χωρίς απώλεια ακρίβειας.
- Ο controller επιστρέφει μία απλή HTTP 2xx απόκριση στο TTS, ώστε ο network server να θεωρήσει το uplink παραδόθηκε και να μην το επανεκπέμψει.

Με αυτόν τον τρόπο, η Spring Boot εφαρμογή λειτουργεί ως «γέφυρα» ανάμεσα στο LoRaWAN επίπεδο και στην επίμονη αποθήκευση (PostgreSQL), παρέχοντας μία καθαρή και επεκτάσιμη διεπαφή για κατανάλωση των δεδομένων από το frontend.

### 6.2.3 REST API προς την React εφαρμογή

Πέρα από το webhook, το backend εκθέτει και ένα σύνολο από REST endpoints για ανάγνωση δεδομένων από την React εφαρμογή. Ενδεικτικά:

- **POST /api/auth/login:** δέχεται username/password και, σε επιτυχία, επιστρέφει JWT για χρήση στις επόμενες κλήσεις.
- **POST /api/auth/logout:** προαιρετικό endpoint για συνέπεια ροής. Στην πράξη το logout γίνεται client-side με διαγραφή του JWT. Επιστρέφει HTTP 200 OK με μήνυμα επιτυχίας.
- **GET /api/data/meter-data:** επιστρέφει λίστα με όλους τους μετρητές (σύνοψη/μεταδεδομένα).
- **GET /api/data/meter-data/sensor-data/{meterId}:** επιστρέφει όλες τις εγγραφές μέτρησης που αντιστοιχούν στον συγκεκριμένο μετρητή.
- **GET /api/data/meter-data/sensor-data/{meterId}/{sensorId}:** επιστρέφει τις εγγραφές του συγκεκριμένου καναλιού/αισθητήρα του μετρητή.
- **GET /api/health:** απλό health-check για διαθεσιμότητα υπηρεσίας.

Τα ουσιώδη endpoints (πλην login και health) προστατεύονται με JWT. Ο client οφείλει να στέλνει Authorization: Bearer <token>, διαφορετικά λαμβάνει HTTP 401.

## 6.3 Frontend: React διεπαφή χρήστη

Το frontend της εφαρμογής υλοποιείται με React και TypeScript και οργανώνεται ως SPA με διακριτές σελίδες/οθόνες: Login, Dashboard και συστατικά (components) για την αναπαράσταση των μετρήσεων. Για τη γραφική απεικόνιση χρησιμοποιείται η βιβλιοθήκη react-chartjs-2 μαζί με Chart.js, ενώ για την επικοινωνία με το backend χρησιμοποιείται Axios για HTTP JSON κλήσεις.

### 6.3.1 Δομή και βασικά components

Η δομή του frontend περιλαμβάνει ενδεικτικά τα εξής:

- **App.tsx:** Κεντρικό component της εφαρμογής, στο οποίο ορίζονται οι διαδρομές (routes) προς τη σελίδα Login και προς το Dashboard (με προστασία μέσω PrivateRoute ώστε να απαιτείται έγκυρο JWT).
- **Login component:** Απλή φόρμα με πεδία username/password, που καλεί το POST /api/auth/login. Σε επιτυχία αποθηκεύει το token σε localStorage ή sessionStorage και ανακατευθύνει στο Dashboard.

- **Dashboard component:** Βασική οθόνη εποπτείας. Περιλαμβάνει επιλογείς χρονικού διαστήματος (π.χ. ημερομηνία από/έως), επιλογή μετρητή ή φάσης, κουμπί Refresh που επαναφέρει τα δεδομένα με νέα κλήση στο backend, καθώς και ένα ή περισσότερα διαγράμματα/πίνακες.
- **SensorChart.tsx:** Component για την απεικόνιση των χρονοσειρών των μετρήσεων. Ανάλογα με την επιλογή του χρήστη, εμφανίζει line, bar ή pie chart, με άξονα χρόνου (ημερομηνία/ώρα timestamp) στον οριζόντιο άξονα και την επιλεγμένη φυσική ποσότητα (τάση, ρεύμα, ισχύς, ενέργεια, συχνότητα, συντελεστής ισχύος) στον κατακόρυφο.
- **SensorData.tsx:** Πίνακας με τις ωμές μετρήσεις, για αναλυτικό έλεγχο. Κάθε γραμμή περιλαμβάνει timestamp, sensorId (φάση), τάση, ρεύμα, ισχύ, ενέργεια, συχνότητα και συντελεστή ισχύος.
- **Separator.tsx** και βοηθητικά components: Απλά βοηθητικά στοιχεία για τη διάταξη (layout) και την οπτική ομαδοποίηση του περιεχομένου.

Η δομή των TypeScript interfaces (π.χ. SensorData) ευθυγραμμίζεται με το JSON που επιστρέφει το backend, ώστε ο μετασχηματισμός των δεδομένων να είναι ευθύγραμμος: τα πεδία current, voltage, power κ.λπ. προέρχονται απευθείας από τις οντότητες της βάσης δεδομένων.

### 6.3.2 Διαχείριση JWT και κλήσεις προς το REST API

Κατά την είσοδο του χρήστη, το Login component στέλνει HTTP POST στη διαδρομή /api/auth/login. Αν τα διαπιστευτήρια είναι σωστά, το backend επιστρέφει ένα JWT token (τυπικά στη μορφή header.payload.signature). Το frontend αποθηκεύει το token και σε κάθε επόμενη κλήση προς κάποιο προστατευμένο endpoint, προσθέτει στην επικεφαλίδα Authorization την τιμή Bearer <token>.

Η βιβλιοθήκη Axios ρυθμίζεται έτσι ώστε να περιλαμβάνει αυτόματα την κεφαλίδα σε όλες τις κλήσεις προς /api/.... Αν το backend επιστρέψει 401 Unauthorized (π.χ. λόγω ληγμένου token), η εφαρμογή μπορεί ανακατευθύνει τον χρήστη στη σελίδα Login και εμφανίζει σχετικό μήνυμα λάθους.

### 6.3.3 Γραφική αναπαράσταση και επιλογές χρήστη

Η οθόνη Dashboard δίνει τη δυνατότητα στον χρήστη να:

- επιλέξει το χρονικό διάστημα προβολής (π.χ. τελευταία ώρα, τελευταία ημέρα, προσαρμοσμένο διάστημα),
- φιλτράρει τα δεδομένα ανά μετρητή ή ανά φάση (π.χ. μόνο sensor1 ή όλες οι φάσεις ταυτόχρονα),
- επιλέξει ποια φυσική ποσότητα θα απεικονιστεί (τάση, ρεύμα, ισχύς, ενέργεια, συχνότητα, power factor),

- αλλάξει τύπο διαγράμματος (line, bar, pie) ανάλογα με την πληροφορία που θέλει να αναδείξει,
- επαναφορτώσει τα δεδομένα με κουμπί Refresh, ώστε να εμφανιστούν τα πιο πρόσφατα uplinks.

Στην επόμενη ενότητα παρατίθενται στιγμιότυπα (screenshots) του UI της εφαρμογής, όπου επεξηγούνται αναλυτικά οι επιλογές, οι αλληλεπιδράσεις του χρήστη και παραδείγματα πραγματικών δεδομένων από το σύστημα τριφασικής μέτρησης.

## 6.4 Εκκίνηση εφαρμογής και παρουσίαση λειτουργειών

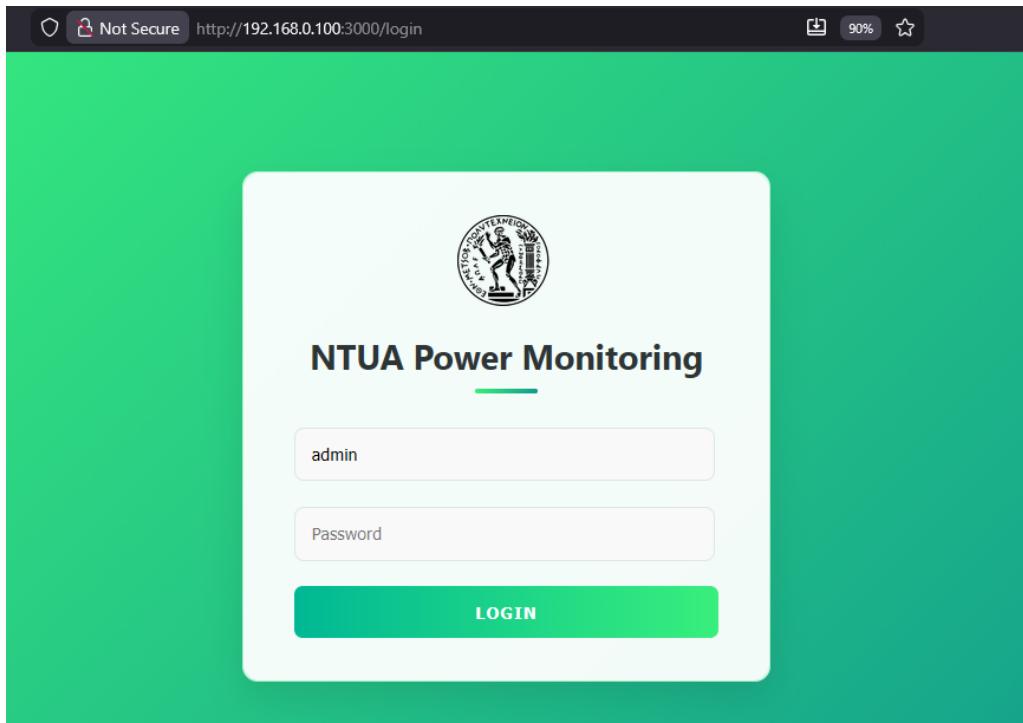
Αρχικά, ρυθμίζουμε ένα docker-compose.yml αρχείο (και τα αντίστοιχα Dockerfile αρχεία) προκειμένου να τρέξουμε το backend και το frontend μέρος της διαδικτυακής εφαρμογής μας μέσω docker:

```

1  services:
2    postgres:
3      image: postgres:13
4      container_name: power-monitoring-postgres
5      environment:
6        POSTGRES_DB: power_monitoring
7        POSTGRES_USER: postgres
8        POSTGRES_PASSWORD: password
9      volumes:
10        - postgres-data:/var/lib/postgresql/data
11      ports:
12        - "5433:5432"
13
14  app:
15    image: power-monitoring-app
16    container_name: power-monitoring-app
17    depends_on:
18      - postgres
19    environment:
20      SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/power_monitoring
21      SPRING_DATASOURCE_USERNAME: postgres
22      SPRING_DATASOURCE_PASSWORD: password
23      SPRING_PROFILES_ACTIVE: dev
24    ports:
25      - "8080:8080"
26
27  frontend:
28    image: power-monitoring-frontend
29    container_name: power-monitoring-frontend
30    depends_on:
31      - app
32    ports:
33      - "3000:80"
34
35  volumes:
36    postgres-data:

```

Στη συνέχεια τρέχουμε την εντολή `docker compose up` για να εκκινήσουμε τα container. Αφότου ενεργοποιηθούν, ανοίγουμε τον φυλλομετρητή της επιλογής μας και συνδεόμαστε στην διεύθυνση <http://192.168.0.100:3000/>. Η εφαρμογή μας ανακατευθύνει στην σελίδα login:



Εικόνα 6.2: Σελίδα Login της διαδικτυακής εφαρμογής.

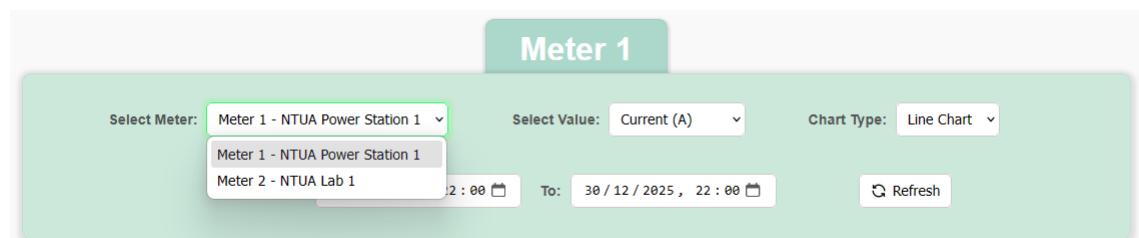
Συνδεόμαστε με τα διαπιστευτήρια που έχουμε αποθηκεύσει στη βάση του backend:

**Username:** admin

**Password:** password

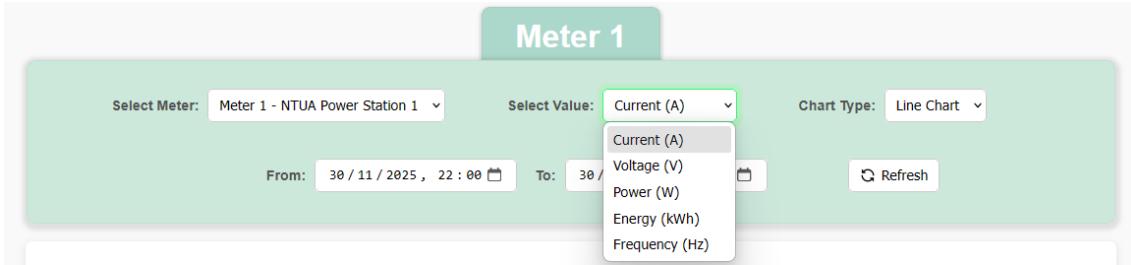
Αμέσως μετά φορτώνεται η βασική σελίδα της εφαρμογής (Εικόνα 6.7). Στο πάνω μέρος δεξιά βλέπουμε το κόκκινο κουμπί *Logout* με το οποίο μπορούμε να αποσυνδεθούμε από τον λογαριασμό μας. Ακολούθως στην καρτέλα από κάτω βλέπουμε στην κορυφή να αναγράφεται ο επιλεγμένος προω επισκόπηση μετρητής και ακριβώς από κάτω τα φίλτρα για την επιλογή των δεδομένων προς αναπαράσταση. Τα φίλτρα σίναι τα εξής:

- **Select Meter:** Επιλογή του μετρητή που θα προβληθεί.



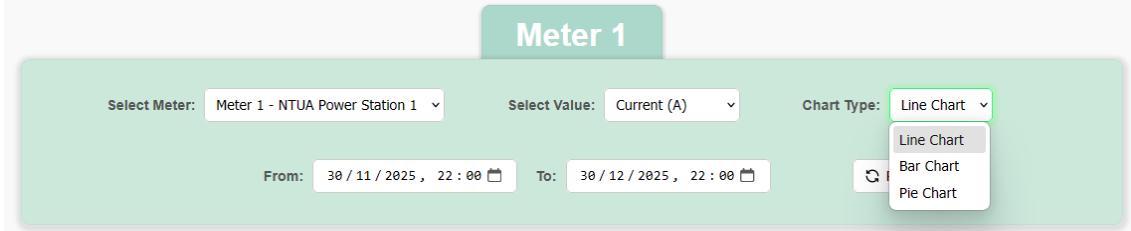
Εικόνα 6.3: Επιλογή μετρητή προς απεικόνιση.

- **Select Value:** Επιλογή μετρητικής ποσότητας που θα απεικονιστεί στο γράφημα.



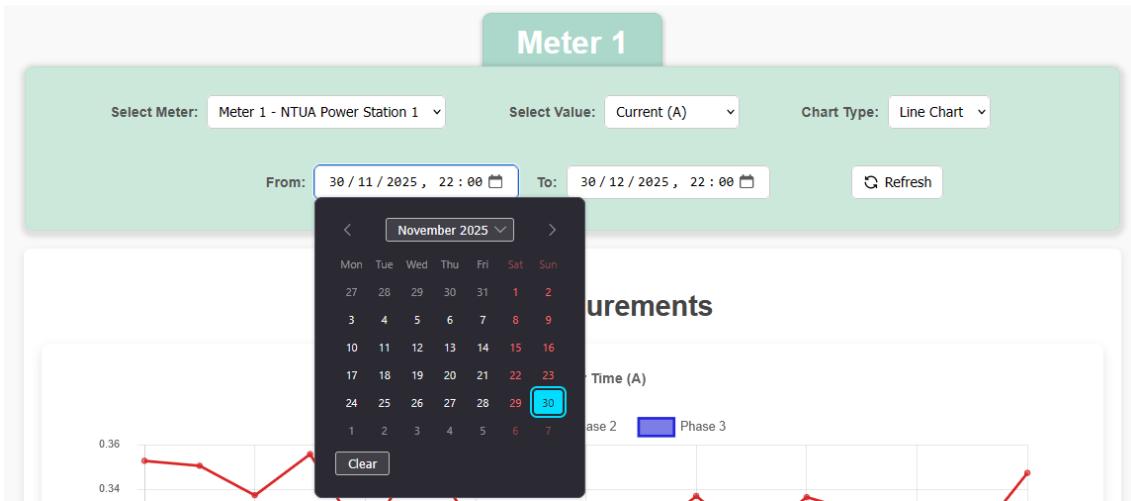
Εικόνα 6.4: Επιλογή μετρητή προς απεικόνιση.

- **Chart Type:** Επιλογή μορφής διαγράμματος για την απεικόνιση (Line Chart, Bar Chart, Pie Chart).



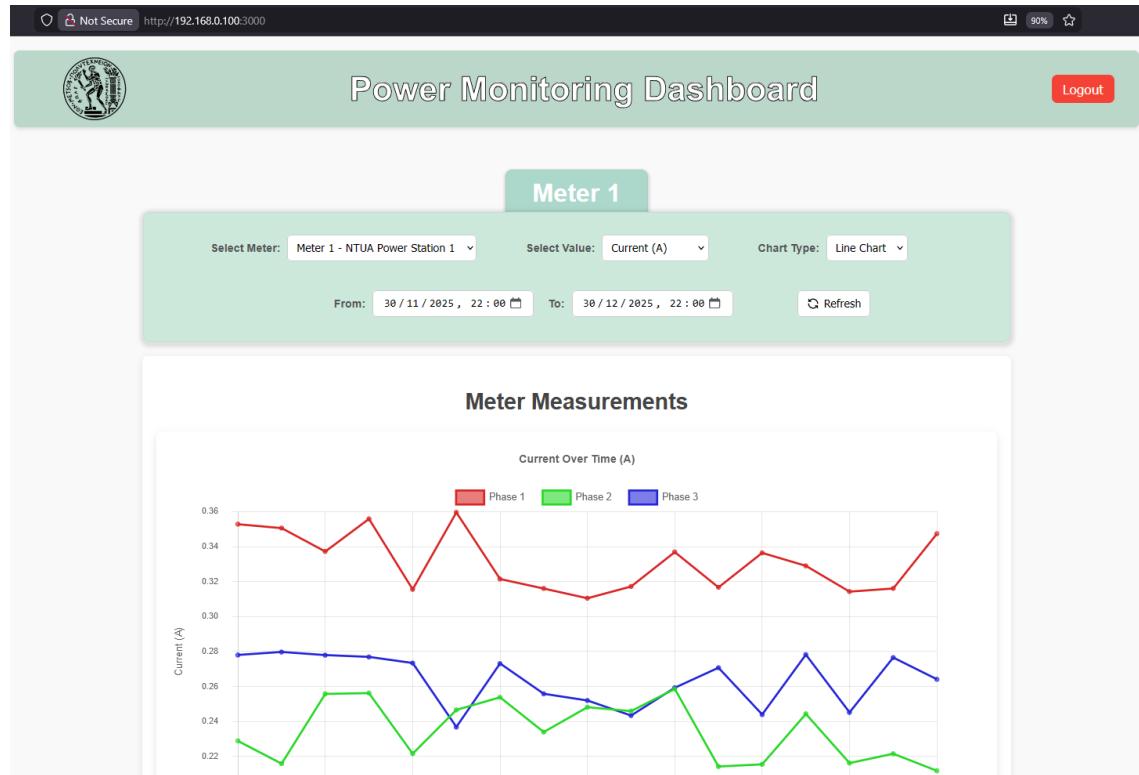
Εικόνα 6.5: Επιλογή μορφής διαγράμματος.

- **From - ημερομηνία/ώρα:** Ορισμός κάτω ορίου του χρονικού διαστήματος των δεδομένων με date-time picker.



Εικόνα 6.6: Επιλογή κάτω ορίου του χρονικού διαστήματος των δεδομένων.

- **To - ημερομηνία/ώρα:** Όμοια, ορισμός άνω ορίου του χρονικού διαστήματος των δεδομένων με date-time picker.
- **Refresh:** Εκτέλεση αναζήτησης με βάση τις τρέχουσες επιλογές (μετρητής, ποσότητα, εύρος ημερομηνιών, τύπος γραφήματος) και ανανέωση του γραφήματος.



Εικόνα 6.7: Βασική σελίδα (Dashboard) της διαδικτυακής εφαρμογής.

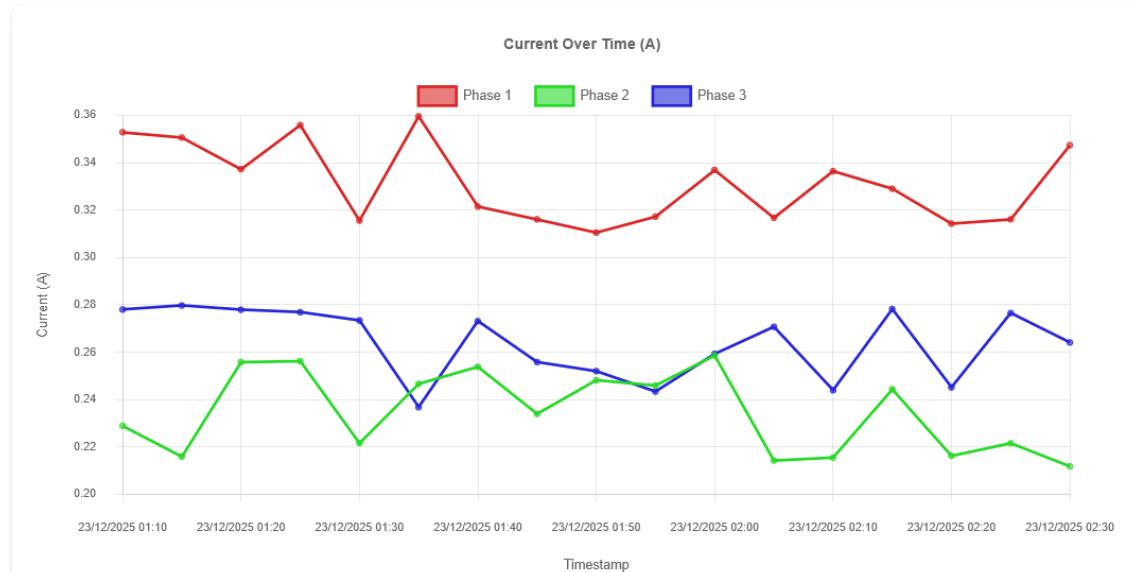
Κατεβαίνοντας πιο κάτω στην σελίδα βρίσκεται επίσης ένας πίνακας που παρουσιάζει αναλυτικά όλες τις μετρήσεις του συγκεκριμένου χρονικού διαστήματος που έχουμε ορίσει στα φίλτρα, για τον δεδομένο μετρητή:

Phase	Current (A)	Energy (kWh)	Frequency (Hz)	Power (W)	Power Factor	Voltage (V)	Timestamp
Phase 1	0.35	0.002	49.91	76.20	0.99	232.28	23/12/2025 01:10:00
Phase 2	0.23	0.001	50.02	53.89	0.98	232.89	23/12/2025 01:10:00
Phase 3	0.28	0.002	49.91	57.57	0.98	232.85	23/12/2025 01:10:00
Phase 1	0.35	0.002	49.99	77.17	0.99	232.36	23/12/2025 01:15:00
Phase 2	0.22	0.001	50.02	55.78	0.98	232.24	23/12/2025 01:15:00
Phase 3	0.28	0.002	49.91	60.58	0.98	232.12	23/12/2025 01:15:00
Phase 1	0.34	0.002	49.95	78.04	0.99	232.15	23/12/2025 01:20:00
Phase 2	0.26	0.001	50.06	53.32	0.98	232.23	23/12/2025 01:20:00
Phase 3	0.28	0.002	49.92	60.91	0.98	232.86	23/12/2025 01:20:00
Phase 1	0.36	0.002	49.94	77.35	0.99	232.03	23/12/2025 01:25:00
Phase 2	0.26	0.001	49.96	51.02	0.98	232.73	23/12/2025 01:25:00
Phase 3	0.28	0.002	49.93	60.54	0.98	232.34	23/12/2025 01:25:00
Phase 1	0.32	0.002	50.06	79.09	0.99	232.79	23/12/2025 01:30:00

Εικόνα 6.8: Πίνακας με την αναλυτική παρουσίαση των δεδομένων.

## Chart Types

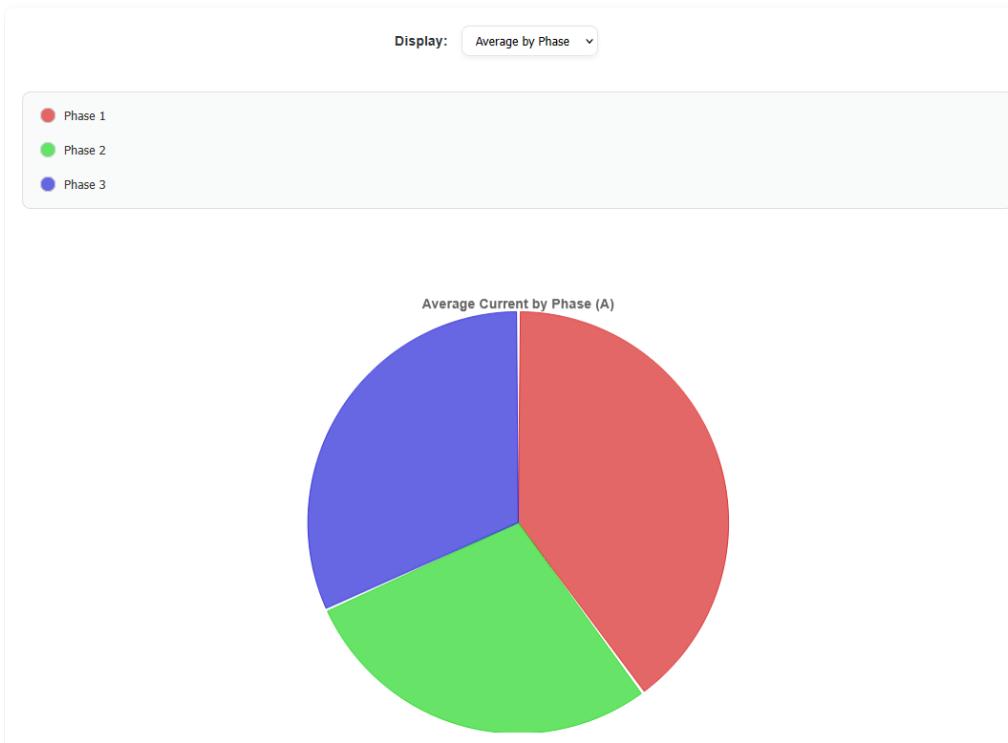
### Meter Measurements

Εικόνα 6.9: *Line Chart*.

### Meter Measurements

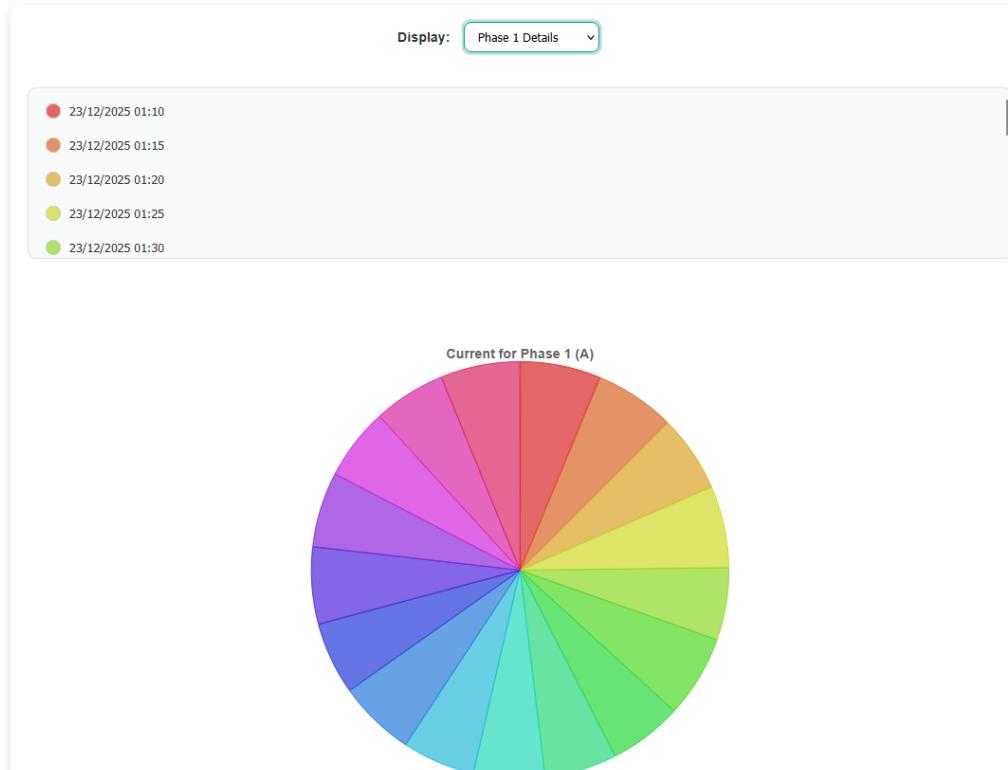
Εικόνα 6.10: *Bar Chart*.

## Meter Measurements



Εικόνα 6.11: *Pie Chart (Average by phase).*

## Meter Measurements



Εικόνα 6.12: *Pie Chart.*

## 6.5 Ολοκλήρωση εγκατάστασης στο Raspberry Pi: ενοποιημένη εκκίνηση υπηρεσιών και auto-start στο boot

Αφού ολοκληρώθηκε η ανάπτυξη (deployment) όλων των υποσυστημάτων στο LoRaWAN gateway (Raspberry Pi), στόχος είναι η **εκκίνηση/τερματισμός με ένα βήμα** και η **αυτόματη εκκίνηση μετά από επανεκκίνηση** της συσκευής. Στον τελικό κόμβο συνυπάρχουν τα εξής Docker σύνολα:

- **The Things Stack (TTS):** Docker compose στο /home/loragw/TheThingsStack.
- **LoRa Basics Station:** Docker compose στο /home/loragw/BasicStation.
- **Power Monitoring Web App:** PostgreSQL + Spring Boot backend + React frontend μέσω Docker compose στο /home/loragw/power-monitoring-backend.

### 6.5.1 Σενάρια start/stop: εκκίνηση όλων των υπηρεσιών «με ένα κλικ»

Η τελική ιεραρχία των φακέλων στο raspberry pi είναι:

```
loragw@loragateway:~$ ls
BasicStation      power-monitoring-backend  power-monitoring-frontend
TheThingsStack
```

Για να αυτοματοποιηθεί η διαδικασία, υλοποιήθηκαν δύο bash scripts στον φάκελο /home/loragw:

- **start\_services.sh:** εκκινεί TTS → περιμένει 5 δευτερόλεπτα → εκκινεί BasicStation → εκκινεί web app.
- **stop\_services.sh:** τερματίζει σε αντίστροφη σειρά (web app → BasicStation → TTS) ώστε να ελαχιστοποιηθούν ασυνέπειες και «օρφανές» εξαρτήσεις.

Ενδεικτικά, το start\_services.sh έχει τη μορφή:

```
1 #!/bin/bash
2 # Define paths and color codes
3 BASICSTATION_DIR="/home/loragw/BasicStation"
4 THINGSTACK_DIR="/home/loragw/TheThingsStack"
5 WEBAPP_DIR="/home/loragw/power-monitoring-backend"
6 GREEN='\033[0;32m'
7 YELLOW='\033[0;33m'
8 RED='\033[0;31m'
9 NC='\033[0m' # No color
10
11 # Start The Things Stack services
12 echo -e "${YELLOW}Starting The Things Stack Docker services...${NC}"
```

```

13 if [ -d "$THINGSTACK_DIR" ]; then
14   (cd "$THINGSTACK_DIR" && sudo docker compose up -d)
15   if [ $? -eq 0 ]; then
16     echo -e "${GREEN}The Things Stack started successfully.${NC}"
17   else
18     echo -e "${RED}Failed to start The Things Stack.${NC}"
19     exit 1
20   fi
21 else
22   echo -e "${RED}Directory $THINGSTACK_DIR does not exist.${NC}"
23   exit 1
24 fi
25
26 # Wait for 5 seconds before starting BasicStation
27 echo -e "${YELLOW}Waiting 5 seconds to ensure The Things Stack is up...${NC}"
28 sleep 5
29
30 # Start BasicStation services
31 echo -e "${YELLOW}Starting BasicStation Docker services...${NC}"
32 if [ -d "$BASICSTATION_DIR" ]; then
33   (cd "$BASICSTATION_DIR" && sudo docker compose up -d)
34   if [ $? -eq 0 ]; then
35     echo -e "${GREEN}BasicStation started successfully.${NC}"
36   else
37     echo -e "${RED}Failed to start BasicStation.${NC}"
38     exit 1
39   fi
40 else
41   echo -e "${RED}Directory $BASICSTATION_DIR does not exist.${NC}"
42   exit 1
43 fi
44
45 # --- Start Web App (backend + frontend + its postgres) ---
46 echo -e "${YELLOW}Starting Power Monitoring Web App services...${NC}"
47 if [ -d "$WEBAPP_DIR" ]; then
48   (cd "$WEBAPP_DIR" && sudo docker compose up -d)
49   if [ $? -eq 0 ]; then
50     echo -e "${GREEN}Power Monitoring Web App started successfully.${NC}"
51   else
52     echo -e "${RED}Failed to start Power Monitoring Web App.${NC}"
53     exit 1
54   fi
55 else
56   echo -e "${RED}Directory $WEBAPP_DIR does not exist.${NC}"
57   exit 1
58 fi
59
60 echo -e "${GREEN}All services started successfully.${NC}"

```

Ενώ ο τερματισμός γίνεται ως εξής:

```

1 #!/bin/bash
2 # Define paths and color codes
3 BASICSTATION_DIR="/home/loragw/BasicStation"
4 THINGSTACK_DIR="/home/loragw/TheThingsStack"
5 WEBAPP_DIR="/home/loragw/power-monitoring-backend"
6 GREEN='\033[0;32m'
7 YELLOW='\033[0;33m'
8 RED='\033[0;31m'
9 NC='\033[0m' # No color
10
11 # --- Stop Web App (backend + frontend + its postgres) ---
12 echo -e "${YELLOW}Stopping Power Monitoring Web App services...${NC}"
13 if [ -d "$WEBAPP_DIR" ]; then
14     (cd "$WEBAPP_DIR" && sudo docker compose down)
15     if [ $? -eq 0 ]; then
16         echo -e "${GREEN}Power Monitoring Web App stopped successfully.${NC}"
17     else
18         echo -e "${RED}Failed to stop Power Monitoring Web App.${NC}"
19     fi
20 else
21     echo -e "${RED}Directory $WEBAPP_DIR does not exist.${NC}"
22 fi
23
24 # Stop BasicStation services
25 echo -e "${YELLOW}Stopping BasicStation Docker services...${NC}"
26 if [ -d "$BASICSTATION_DIR" ]; then
27     (cd "$BASICSTATION_DIR" && sudo docker compose down)
28     if [ $? -eq 0 ]; then
29         echo -e "${GREEN}BasicStation stopped successfully.${NC}"
30     else
31         echo -e "${RED}Failed to stop BasicStation.${NC}"
32         exit 1
33     fi
34 else
35     echo -e "${RED}Directory $BASICSTATION_DIR does not exist.${NC}"
36     exit 1
37 fi
38
39 # Wait for 5 seconds before stopping The Things Stack
40 echo -e "${YELLOW}Waiting 5 seconds before stopping The Things Stack...${NC}"
41 sleep 5
42
43 # Stop The Things Stack services
44 echo -e "${YELLOW}Stopping The Things Stack Docker services...${NC}"
45 if [ -d "$THINGSTACK_DIR" ]; then
46     (cd "$THINGSTACK_DIR" && sudo docker compose down)
47     if [ $? -eq 0 ]; then

```

```
48     echo -e "${GREEN}The Things Stack stopped successfully.${NC}"
49 else
50     echo -e "${RED}Failed to stop The Things Stack.${NC}"
51     exit 1
52 fi
53 else
54     echo -e "${RED}Directory $THINGSTACK_DIR does not exist.${NC}"
55     exit 1
56 fi
57
58 echo -e "${GREEN}All services stopped successfully.${NC}"
```

Για χειροκίνητη εκτέλεση, τα αρχεία γίνονται εκτελέσιμα ως εξής:

```
chmod +x /home/loragw/start_services.sh /home/loragw/stop_services.sh
```

και μπορύμε να τα τρέξουμε με την εντολή:

```
./start_services.sh
```

```
loragw@loragateway:~$ chmod +x start_services.sh stop_services.sh
loragw@loragateway:~$ ./start_services.sh
Starting The Things Stack Docker services...
[+] Running 4/4
  ✓ Network thethingsstack_default  Created      0.1s
  ✓ Container redis                Started     1.2s
  ✓ Container postgres              Started     1.1s
  ✓ Container stack                Started     1.5s

The Things Stack started successfully.

Waiting 5 seconds to ensure The Things Stack is up...
Starting BasicStation Docker services...
[+] Running 1/1
  ✓ Network thethingsstack_default  Started      0.1s

BasicStation started successfully.

Starting Power Monitoring Web App services...
[+] Running 4/4
  ✓ Network power-monitoring-backend_default  Created      0.1s
  ✓ Container power-monitoring-postgres       Started     1.1s
  ✓ Container power-monitoring-app            Started     1.1s
  ✓ Container power-monitoring-frontend       Started     1.4s

Power Monitoring Web App started successfully.

All services started successfully.
```

```

loragw@loragateway:~$ ./stop_services.sh
Stopping Power Monitoring Web App services...
[+] Running 4/4
  ✓ Network power-monitoring-backend_default  Removed      0.6s
  ✓ Container power-monitoring-postgres       Removed     10.6s
  ✓ Container power-monitoring-app           Removed      0.6s
  ✓ Container power-monitoring-frontend      Removed      0.4s
Power Monitoring Web App stopped successfully.

Stopping BasicStation Docker services...
[+] Running 1/1
  ✓ Network thethingsstack_default  Removed      10.4s
BasicStation stopped successfully.

Waiting 5 seconds before stopping The Things Stack...
Stopping The Things Stack Docker services...
[+] Running 4/4
  ✓ Network thethingsstack_default  Removed      11.5s
  ✓ Container redis                Removed      0.7s
  ✓ Container postgres              Removed      0.9s
  ✓ Container stack                Removed      0.4s
The Things Stack stopped successfully.

All services stopped successfully.

```

### 6.5.2 Αυτόματη εκκίνηση στο boot με systemd

Για να εκκινούν **αυτόματα** οι υπηρεσίες σε κάθε επανεκκίνηση του Raspberry Pi, δημιουργείται μία systemd unit η οποία: (α) περιμένει να είναι διαθέσιμο το δίκτυο και ο Docker daemon, και (β) καλεί τα start/stop scripts. Το αρχείο /etc/systemd/system/power-stack.service δημιουργείται ως εξής:

```
sudo nano /etc/systemd/system/power-stack.service
```

και το περιεχόμενό του είναι:

```

1 [Unit]
2 Description=Start LoRaWAN services (TTS, BasicStation, Power Monitoring app)
3 After=network-online.target docker.service
4 Wants=network-online.target docker.service
5
6 [Service]
7 Type=oneshot
8 WorkingDirectory=/home/loragw
9 ExecStart=/home/loragw/start_services.sh
10 ExecStop=/home/loragw/stop_services.sh
11 RemainAfterExit=yes

```

```
12 StandardOutput=journal  
13 StandardError=journal  
14  
15 [Install]  
16 WantedBy=multi-user.target
```

Έπειτα ενεργοποιείται στο boot και εκκινεί άμεσα με τις εντολές:

- Επαναφορτώνει τις μονάδες του systemd ώστε να αναγνωριστούν τυχόν αλλαγές/νέες υπηρεσίες (.service φιλες).

```
sudo systemctl daemon-reload
```

- Ενεργοποιεί την υπηρεσία για αυτόματη εκκίνηση στο boot (δημιουργεί τα απαραίτητα symlinks).

```
sudo systemctl enable power-stack.service
```

- Εκκινεί άμεσα την υπηρεσία χωρίς να απαιτείται επανεκκίνηση του συστήματος.

```
sudo systemctl start power-stack.service
```

Για έλεγχο κατάστασης και ανάγνωση καταγραφών:

```
sudo systemctl status power-stack.service  
sudo journalctl -u power-stack.service -e
```

Με τη συγκεκριμένη ρύθμιση, το Raspberry Pi λειτουργεί ως πλήρως αυτόνομος gateway/server κόμβος: μετά από reboot εκκινούν αυτόματα το The Things Stack, το LoRa Basics Station και η διαδικτυακή εφαρμογή εποπτείας, χωρίς να απαιτείται χειροκίνητη παρέμβαση.

## Μέρος **III**

### Επίλογος



## Κεφάλαιο 7

# Συμπεράσματα και Μελλοντικές Κατευθύνσεις

### 7.1 Συμπεράσματα

Η παρούσα εργασία υλοποίησε ένα ολοκληρωμένο σύστημα τριφασικής μέτρησης και απομακρυσμένης εποπτείας ενέργειας, βασισμένο στο οικοσύστημα LoRa/LoRaWAN, με gateway τύπου Raspberry Pi 4 Model B και συγκεντρωτή iC880A-SPI, ενώ ως Network Server αξιοποιήθηκε το The Things Stack και ως packet forwarder το LoRa Basics Station. Στο ανώτερο επίπεδο της λύσης αναπτύχθηκε πλήρης web εφαρμογή με Spring Boot/PostgreSQL για συλλογή και επίμονη αποθήκευση μετρήσεων και React frontend για αλληλεπιδραστική οπτικοποίηση. Η αρχική ερευνητική στοχοθεσία αφορούσε την πρακτική διερεύνηση της αξιοπιστίας, της επεκτασιμότητας και της χρησικότητας του LoRaWAN σε περιβάλλοντα υποσταθμών. Τα αποτελέσματα επιβεβαιώνουν ότι, με προσεκτική αρχιτεκτονική και ορθή παραμετροποίηση, το πρωτόκολλο και τα συνοδευτικά εργαλεία ανταποκρίνονται επαρκώς στις απαιτήσεις που τέθηκαν.

Καθοριστική συνιστώσα της επιτυχίας υπήρξε η ενοποίηση όλης της στοίβας στο ίδιο gateway. Η ενορχήστρωση The Things Stack, LoRa Basics Station και της εφαρμογής παρακολούθησης μέσω Docker/Compose, πλαισιωμένη από systemd units για αυτόματη εκκίνηση, οδήγησε σε μια λειτουργική «one-touch» υλοποίηση: το σύστημα εκκινεί, συγχρονίζει τις απαραίτητες υπηρεσίες, δέχεται uplinks, τα δρομολογεί στο webhook του backend και προσφέρει άμεσα διαθέσιμα διαγράμματα στο dashboard. Η προσέγγιση αυτή απλοποιεί τη διάθεση (deployment) σε απομακρυσμένα σημεία, μειώνει τον λειτουργικό φόρτο και καθιστά την εγκατάσταση επαναλήψιμη με ελάχιστα βήματα.

Επιπλέον, η ροή δεδομένων *άκρου → δικτύου → εφαρμογής* αποδείχθηκε αξιόπιστη. Η διασύνδεση HTTP webhook επέτρεψε καθαρή αποσύνδεση της δικτυακής στοίβας από τη λογική της εφαρμογής, προσφέροντας ιχνηλασιμότητα ανά uplink, σαφή χειρισμό σφαλμάτων και απλή επαλήθευση της παραλαβής. Η επίμονη αποθήκευση σε PostgreSQL με κατάλληλο σχήμα και η REST διεπαφή ανάκτησης δεδομένων διευκόλυνεν την παραγωγή διαδραστικών γραφημάτων (π.χ. ρεύμα, τάση, ισχύς, ενέργεια, συχνότητα) με φίλτρα σε μετρητή, μέγεθος και χρονικό διάστημα. Η συνολική εμπειρία χρήσης στο frontend κατέδειξε ότι, ακόμη και σε μη ιδανικές συνθήκες δικτύου, η λύση παρέχει συνεκτική εικόνα λειτουργίας του συστήματος σε σχεδόν πραγματικό χρόνο.

Ιδιαίτερη αξία είχαν οι εργαστηριακές δοκιμές με ωμικά φορτία (λαμπτήρες πυρακτώσεως), όπου καταγράφηκαν συντελεστές ισχύος κοντά στη μονάδα και τιμές ισχύος που συμφω-

νούν με τη θεωρητική προσέγγιση  $P \approx V \cdot I$ , λαμβάνοντας υπόψη τις μικρές αποκλίσεις τάσης και τις ανοχές του εξοπλισμού. Οι μετρήσεις ενέργειας επιβεβαίωσαν επίσης τη συνέπεια του συστήματος ως προς τη χρονική συσσώρευση (kWh). Μέσα από αυτήν τη διαδικασία, αξιολογήθηκε πρακτικά η ακρίβεια του μετρητικού υποσυστήματος και η ακεραιότητα του πακεταρίσματος/μεταφοράς των δεδομένων μέχρι την τελική απεικόνιση.

Η διαδικασία εγκατάστασης τεκμηριώθηκε πλήρως: από τη δημιουργία headless εικόνας του Raspberry Pi OS, την ενεργοποίηση της διεπαφής SPI και τη συνδεσμολογία με τον συγκεντρωτή, μέχρι την κλωνοποίηση των αποθετηρίων, την προσαρμογή docker-compose.yml και τη ρύθμιση των systemd units για εκκίνηση/τερματισμό του συνόλου. Η τεκμηρίωση αυτή αποτελεί πρακτικό οδηγό αναπαραγωγής της λύσης και μειώνει το «γνωσιακό χρέος» για μελλοντικούς συντηρητές ή επεκτάτες.

Ωστόσο, υπάρχουν και ορισμένοι περιορισμοί που πρέπει να αναγνωριστούν. Η θεμελιώδης ανταλλαγή στο LoRaWAN ανάμεσα σε ρυθμό μετάδοσης, εμβέλεια και duty-cycle συνεπάγεται ότι οι παραμετροί SF/DR πρέπει να επιλέγονται με προσοχή ανάλογα με το περιβάλλον, την πυκνότητα κόμβων και τις απαιτήσεις latency. Οι δοκιμές πραγματοποιήθηκαν με περιορισμένο αριθμό μετρητών και με ωμικά φορτία, άρα το φάσμα παρεμβολών και ο θόρυβος ενός «βαρέος» βιομηχανικού περιβάλλοντος δεν διερευνήθηκαν πλήρως. Τέλος, η συνειδητή επιλογή συγκέντρωσης όλων των υπηρεσιών στον ίδιο κόμβο (single-node hosting) διευκολύνει την εγκατάσταση αλλά δεν αποτυπώνει σενάρια υψηλής διαθεσιμότητας ή οριζόντιας κλιμάκωσης.

Συνοψίζοντας, η εργασία αποδεικνύει ότι μια στοχευμένη, ανοιχτού κώδικα στοίβα LoRaWAN μπορεί να υλοποιήσει μια πρακτική λύση «άκρο-σε-άκρο» για παρακολούθηση ενεργειακών μεγεθών, με χαμηλό κόστος, υψηλή επαναληψιμότητα εγκατάστασης και επαρκή αξιοποιητική για τις ανάγκες εποπείας υποσταθμών και παρόμοιων βιομηχανικών εφαρμογών.

## 7.2 Μελλοντικές κατευθύνσεις

Με βάση τα παραπάνω ευρήματα, διαφαίνονται πολλαπλές επεκτάσεις που μπορούν να ωριμάσουν περαιτέρω τη λύση. Πρώτη προτεραιότητα είναι η κλιμάκωση και η ανθεκτικότητα. Η μεταφορά του The Things Stack και της βάσης δεδομένων σε ξεχωριστούς κόμβους, η υιοθέτηση Docker Swarm ή Kubernetes και η εισαγωγή αντιγράφων (replicas) για την PostgreSQL με στρατηγικές backup/restore θα επιτρέψουν ορισμό στόχων RPO/RTO και επιχειρησιακή συνέχεια σε περιπτώσεις αστοχίας. Παράλληλα, η ενσωμάτωση παρακολούθησης πόρων και εφαρμογών με Prometheus/Grafana θα προσφέρει ορατότητα, έγκαιρα προειδοποιητικά σήματα και μετρήσιμη βάση για βελτιστοποίηση.

Σε επίπεδο ραδιοζεύχης, αξίζει μια συστηματικότερη μελέτη ποιότητας: συλλογή και ανάλυση RSSI/SNR σε ποικίλες γεωμετρίες εγκατάστασης, δοκιμές με διαφορετικά μήκη-/τύπους ομοαξονικών καλωδίων και κεραιών, καθώς και πειράματα με υψομετρικές διαφοροποιήσεις. Η αξιοποίηση του ADR για δυναμική προσαρμογή του ρυθμού δεδομένων μπορεί να βελτιώσει τον λόγο επιτυχών uplinks σε περιβάλλοντα με άνισο ραδιο-/φορτίο. Στο άκρο της συσκευής, η χρήση τεχνικών συμπίεσης ή αποδοτικότερων σχημάτων κωδικοποίησης τηλεμετρίας θα μείωνε τον χρόνο αέρα (airtime) και τη συνολική κατανάλωση.

Ως προς την ασφάλεια, η λύση μπορεί να σκληρυνθεί «από άκρο σε άκρο». Η χρήση

έγκυρων δημόσιων πιστοποιητικών TLS και η αυστηρή ρύθμιση HTTPS/WSS σε όλες τις διεπαφές, οι πολιτικές ελαχιστοποίησης δικαιωμάτων (π.χ. ξεχωριστοί χρήστες/ρόλοι στη βάση, περιορισμένα scopes στο TTS), καθώς και η τακτική περιστροφή/ανανέωση μυστικών αυξάνουν την εμπιστοσύνη του συστήματος. Στο web επίπεδο, πρακτικές όπως HSTS, προστασία από CSRF και έλεγχος ισχυρών CORS πολιτικών θωρακίζουν την πρόσβαση, ενώ ένα ελαφρύ WAF θα βοηθούσε στην αποτροπή κοινών επιθέσεων.

Σε ό,τι αφορά την αναλυτική αξιοποίηση των δεδομένων, η ενσωμάτωση καναλιών επεξεργασίας (pipelines) για ανίχνευση ανωμαλιών και πρόβλεψη ζήτησης δύναται να μετατρέψει το σύστημα από απλή εποπτεία σε εργαλείο προληπτικής συντήρησης. Μέθοδοι όπως αποσύνθεση χρονοσειρών (STL) ή απλά μονέλα πρόβλεψης, σε συνδυασμό με κανόνες ενεργοποίησης ειδοποιήσεων, μπορούν να εντοπίζουν εγκαίρως ασυνήθιστες συμπεριφορές, διακυμάνσεις συχνότητας ή απότομες αλλαγές συντελεστή ισχύος. Η επέκταση του σχήματος δεδομένων ώστε να συνενώνει μετρήσεις από περισσότερους αισθητήρες (π.χ. θερμοκρασίας, κραδασμών) διευρύνει τον ορίζοντα χρήσης σε περιβάλλοντα όπου απαιτείται πολυπαραμετρική εποπτεία.

Τέλος, σε επίπεδο συμμόρφωσης και εναρμόνισης με πρότυπα, μελλοντική εργασία μπορεί να συμπεριλάβει δοκιμές σύμφωνα με οδηγίες EMC/ασφάλειας και διερεύνηση χαρακτηριστικών LoRaWAN 1.1/1.0.4 που σχετίζονται με roaming ή ιδιωτικές διασυνδέσεις (peering) δικτύων. Η διερεύνηση αυτών των δυνατοτήτων θα καταστήσει τη λύση πιο «βιομηχανικά ώριμη» και έτοιμη για υιοθέτηση σε μεγαλύτερες εγκαταστάσεις, με απαιτήσεις διαλειτουργικότητας και αυστηρά SLA.

Συνολικά, η εργασία απέδειξε τη βιωσιμότητα μιας οικονομικής, ανοιχτού κάθικα λύσης LoRaWAN για μέτρηση και εποπτεία ισχύος. Η προτεινόμενη πορεία εξέλιξης περιλαμβάνει επιχειρησιακή ωρίμανση (κλιμάκωση, ανθεκτικότητα, ασφάλεια), τεχνική βελτίωση της ζεύξης (προσαρμοστικότητα και μετρήσεις ποιότητας) και εμπλουτισμό του πληροφοριακού επιπέδου (ανάλυση/πρόβλεψη, ειδοποιήσεις), έτσι ώστε το σύστημα να μεταβεί από πιλότο επίδειξης σε παραγωγική υποδομή μετρήσεων σε πραγματικές συνθήκες.



## Βιβλιογραφία

---

- [1] *A Rapid Increase in IoT Adoption - Manufacturing IoT in 2023.* [Online]. Available: <https://ubisense.com/a-rapid-increase-in-iot-adoption-manufacturing-iot-in-2023/>, 2023. Accessed: Sep. 16, 2025.
- [2] *Internet of Things (IoT) Statistics and Facts for 2024.* [Online]. Available: <https://www.demandsage.com/internet-of-things-statistics/>. Accessed: Sep. 16, 2025.
- [3] K. Mekki, E. Bajic, F. Chaxel και F. Meyer. *System Design Considerations For Internet Of Things (IoT) With Category-M Devices In LTE Networks.* *ICT Express*, 5(1):11–17, 2019.
- [4] A. Ukovich. *NB-IoT Explained: What Is It, and How Does It Work?* [Online]. Available: <https://www.telit.com/blog/nb-iot-new-cellular-standard-means-business/>, 2019. Accessed: Sep. 16, 2025.
- [5] Zipit Wireless. *LTE Cat-M1 Explained: Pros and Cons of LTE-M for IoT Devices.* [Online]. Available: <https://www.zipitwireless.com/blog/lte-cat-m1-explained-pros-and-cons-of-lte-m-for-iot-devices>, 2023. Accessed: Sep. 16, 2025.
- [6] G. Hosangadi, D. Wang και A. Rao. *System Design Considerations for Internet of Things (IoT) with Category-M Devices in LTE Networks.* *arXiv preprint arXiv:1902.00408*, 2019.
- [7] Semtech Corporation. *LoRa<sup>®</sup> and LoRaWAN<sup>®</sup>: A Technical Overview.* [Online]. Available: <https://www.semtech.com/uploads/technology/LoRa/lora-and-lorawan.pdf>, 2021. Accessed: Sep. 16, 2025.
- [8] S. Mansoor, S. Iqbal, S. M. Popescu, S. L. Kim, Y. S. Chung και J. H. Baek. *Integration of smart sensors and IoT in precision agriculture: trends, challenges and future prospectives.* *Frontiers in Plant Science*, 16, 2025. Accessed: Sep. 16, 2025.
- [9] Saft Batteries. *The impact of communication technology protocol on your IoT application's power consumption.* [Online]. Available: <https://saft.com/en/energizing-iot/impact-communication-technology-protocol-your-iot-application%E2%80%99s-power-consumption>, 2023. Accessed: Sep. 16, 2025.
- [10] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melià-Seguí και T. Watteyne. *Understanding the Limits of LoRaWAN.* *IEEE Communications Magazine*, 55(9):34–40, 2017.

- [11] Semtech Corporation. *LoRa<sup>TM</sup> Modulation Basics*. Application Note AN1200.22, Semtech Corporation, 2015.
- [12] LoRa Alliance Technical Committee, Regional Parameters Workgroup. *LoRaWAN® Regional Parameters v1.0.3revA*. Technical Specification RP002-1.0.3revA, LoRa Alliance, Inc., 2018.
- [13] GSMA. *Mobile IoT Deployment Guide*. [Online]. Available: <https://www.gsma.com/smartmobility/wp-content/uploads/2022/10/Mobile-IOT-Deployment-Guide-October-2022-1-1.pdf>, 2022. Accessed: Sep. 16, 2025.
- [14] LoRa Alliance. *LoRa Alliance Issues 2023 Annual Report Highlighting LoRaWAN Maturity, Robust Adoption, and Diversity of End-to-End Solutions*, 2024. [Online]. Available: <https://lora-alliance.org/lora-alliance-press-release/lora-alliance-issues-2023-annual-report-highlighting-lorawan-maturity-robust-adoption-and-diversity-of-end-to-end-solutions/>. Accessed: Sep. 16, 2025.
- [15] L. Slats και Semtech Corporation. *A Brief History of LoRa®: Three Inventors Share Their Personal Story at The Things Conference*. Semtech Corporate Blog, 2020. [Online]. Available: <https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference>. Accessed: Sep. 16, 2025.
- [16] The Things Network. *LoRaWAN Documentation*. The Things Network, online documentation. [Online]. Available: <https://www.thethingsnetwork.org/docs lorawan/>. Accessed: Sep. 16, 2025.
- [17] T. Rademacher et al. *Analysis of Urban Path Loss Models for LoRaWAN Network Planning*. *International Journal of Advanced Scientific Research and Engineering (IJASRE)*, 4(7):12–21, 2018.
- [18] LoRa Documentation. *LoRa - The Things Network Documentation*. Online documentation, 2024. [Online]. Available: <https://lora.readthedocs.io/en/latest/>. Accessed: Sep. 16, 2025.
- [19] K. Staniec και M. Kowal. *LoRa Performance under Variable Interference and Heavy-Multipath Conditions*. *Wireless Communications and Mobile Computing*, 2018. [Online]. Available: <https://onlinelibrary.wiley.com/doi/full/10.1155/2018/6931083>. Accessed: Sep. 16, 2025.
- [20] Lansitec. *What Are the Factors Affecting LoRaWAN Range in IoT*. Lansitec Blog, 2023. [Online]. Available: <https://www.lansitec.com/blogs/what-are-the-factors-affecting-lorawan-range-in-iot/>. Accessed: Sep. 16, 2025.
- [21] Rohde & Schwarz. *Characterization of LoRa Devices (Rev. 2e)*. Application Note 1MA295, Rohde & Schwarz, 2019.

- [22] A. Baral. *Understanding of LoRa*. Medium blog post, 2020. [Online]. Available: <https://medium.com/@arghyabaran/understanding-of-lora-066d89ed7bf4>. Accessed: Sep. 16, 2025.
- [23] S. Ghoslya. *LoRa: Symbol Generation (All About LoRa and LoRaWAN)*. Blog post All About LoRa and LoRaWAN, 2017. [Online]. Available: <https://www.sghoslyा.com/p/lora-is-chirp-spread-spectrum.html>. Accessed: Sep. 16, 2025.
- [24] M. Janc. *Communication over LoRa - M2M Case Using RA-01*. Sii Blog, 2023. [Online]. Available: <https://sii.pl/blog/en/communication-over-lora-m2m-case-using-ra-01/>. Accessed: Sep. 16, 2025.
- [25] K. Olsson και S. Finnsson. *Exploring LoRa and LoRaWAN: A Suitable Protocol for IoT Weather Stations?* Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2017. [Online]. Available: <https://publications.lib.chalmers.se/records/fulltext/252610/252610.pdf>. Accessed: Sep. 16, 2025.
- [26] M. Hanif και H. H. Nguyen. *Slope-Shift Keying LoRa-Based Modulation*. IEEE Internet of Things Journal, 8(1):211–221, 2021.
- [27] M. Hanif και H. H. Nguyen. *Frequency-Shift Chirp Spread Spectrum Communications With Index Modulation*. IEEE Internet of Things Journal, 8(24):17611–17621, 2021.
- [28] A. Maleki, H. H. Nguyen, E. Bedeer και R. Barton. *A Tutorial on Chirp Spread Spectrum Modulation for LoRaWAN: Basics and Key Advances*. IEEE Open Journal of the Communications Society, 5(99):4578–4612, 2024.
- [29] RF Wireless World. *LoRaWAN Airtime Calculator and LoRa Packet Duration Calculation*. Online Calculator, RF Wireless World, 2025. [Online]. Available: <https://www.rfwireless-world.com/calculators lorawan-airtime-calculator>. Accessed: Sep. 16, 2025.
- [30] LoRa Alliance. *What is LoRaWAN® Specification*. LoRa Alliance, online “About LoRaWAN” page, 2025. [Online]. Available: <https://lora-alliance.org/about-lorawan-old/>. Accessed: Sep. 16, 2025.
- [31] The Things Industries. *The Things Industries - Empowering Global IoT Connectivity*. Company homepage, 2025. [Online]. Available: <https://www.thethingsindustries.com/>. Accessed: Sep. 16, 2025.
- [32] Trend Micro Research. *The Current State of LoRaWAN Security*. Technical Brief, Trend Micro, 2021.
- [33] S. Loukil, L. C. Chaari Fourati, A. Nayyar και K. W. A. Chee. *Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms*. Sensors, 22(10):3717, 2022.
- [34] Semtech Corporation. *LoRaWAN® Device Classes Version 2.0*. Application Note AN1200.87, Semtech Corporation, 2024.

- [35] S. R. Gatla. *Understanding LoRaWAN: The MAC Layer Behind LoRa*. Medium blog post, 2025. [Online]. Available: <https://medium.com/@shiva.cdachyd/understanding-lorawan-the-mac-layer-behind-lora-48c512e2180a>. Accessed: Sep. 16, 2025.
- [36] Semtech. “2 - Sending Messages: Metadata”, *LoRaWAN® 1.0.4 Specification in Depth for End Device Developers*. Semtech Learning Center, online course chapter, 2025. [Online]. Available: <https://learn.semtech.com/mod/book/view.php?id=172&chapterid=104>. Accessed: Aug. 28, 2025.
- [37] LoRa Alliance. *LoRaWAN® Specification v1.1*. Technical Specification TS-001-1.1.0, LoRa Alliance, 2017.
- [38] O. Gimenez και I. Petrov. *Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN (RFC 9011)*. Proposed Standard RFC 9011, Internet Engineering Task Force, 2021.
- [39] The Things Industries. *The Things Stack Documentation*. The Things Industries, online documentation, 2025. [Online]. Available: <https://www.thethingsindustries.com/docs/>. Accessed: Sep. 16, 2025.
- [40] Semtech. *LoRa Basics Station 2.0.6 Documentation*. Online documentation, Semtech, 2022. [Online]. Available: <https://doc.semtech.com/station/>. Accessed: Sep. 16, 2025.
- [41] The Things Industries. *Getting to Know LoRa Basics™ Station*. News article, The Things Industries, 2020. [Online]. Available: <https://www.thethingsindustries.com/news/getting-to-know-lora-basics-station/>. Accessed: Sep. 16, 2025.
- [42] S. Singh. *Let's Learn Docker from Scratch in 2023*. Blog post, DevOps.dev, 2023. [Online]. Available: <https://blog.devops.dev/lets-learn-docker-from-scratch-in-2023-d831629c5497>. Accessed: Sep. 16, 2025.
- [43] Docker, Inc. *Docker Documentation*. Online documentation, 2025. [Online]. Available: <https://docs.docker.com/>. Accessed: Sep. 16, 2025.
- [44] ByteByteGo. *How Does Docker Work?* Online guide, 2024. [Online]. Available: <https://bytebytogo.com/guides/how-does-docker-work/>. Accessed: Sep. 16, 2025.
- [45] Pivotal / VMware / Spring Team. *Spring Boot Documentation*. Online documentation, 2025. [Online]. Available: <https://docs.spring.io/spring-boot/index.html>. Accessed: Sep. 16, 2025.
- [46] IBM. *What Is Java Spring Boot?* Technical article, IBM Think, 2024. [Online]. Available: <https://www.ibm.com/think/topics/java-spring-boot>. Accessed: Sep. 16, 2025.
- [47] Meta / React Team. *React Documentation - Learn React*. Online documentation, 2025. [Online]. Available: <https://react.dev/learn>. Accessed: Sep. 16, 2025.

- [48] PostgreSQL Global Development Group. *PostgreSQL Documentation*. Online documentation, 2025. [Online]. Available: <https://www.postgresql.org/docs/>. Accessed: Sep. 16, 2025.
- [49] The Things Industries. *Raspberry Pi LoRaWAN Gateway Documentation*. Online documentation, The Things Industries, 2025. [Online]. Available: <https://www.thethingsindustries.com/docs/hardware/gateways/models/raspberry-pi/>. Accessed: Sep. 16, 2025.
- [50] Raspberry Pi Ltd. *Raspberry Pi 4 Model B Product Brief*. Τεχνική Αναφορά με αριθμό, Raspberry Pi Ltd., 2025. [Online]. Available: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf>. Accessed: Sep. 16, 2025.
- [51] IMST GmbH / Wireless Solutions. *WiMOD iC880A Datasheet Version 0.50*. Τεχνική Αναφορά με αριθμό, IMST GmbH, 2015. [Online]. Available: [https://shop.imest.de/media/pdf/22/67/a8/iC880A\\_Datasheet\\_V0\\_50.pdf](https://shop.imest.de/media/pdf/22/67/a8/iC880A_Datasheet_V0_50.pdf). Accessed: Sep. 16, 2025.
- [52] Peacefair Electronics. *PZEM-004T V3.0 Datasheet and User Manual*. Technical datasheet / user manual, 2019. [Online]. Available: <https://innovatorsguru.com/wp-content/uploads/2019/06/PZEM-004T-V3.0-Datasheet-User-Manual.pdf>. Accessed: Sep. 16, 2025.
- [53] The Things Network. *The Things Uno Documentation*. Online documentation, 2025. [Online]. Available: <https://www.thethingsnetwork.org/docs/devices/uno/>. Accessed: Sep. 16, 2025.
- [54] Arduino Forum community. *Which pins can be used for SoftwareSerial on Micro/Leonardo*, 2018. [Ονλινε]. Απιλαθλε: <https://forum.arduino.cc/t/ωηιςη-πινσ-ζων-βε-υσεδ-φορ-σοφτωαρε-σεριαλ-ον-μιςρο/518642>. ΡΕ πινς ον ATmega32Y4 βιαρδο: 8,9,10,11,14(ΜΙΣΟ),15(Σ'Κ),16(ΜΟΣΙ).
- [55] Arduino Documentation. *SoftwareSerial Library | Arduino Reference*, 2022. [Ονλινε]. Απιλαθλε: <https://www.arduino.cc/en/Reference/SoftwareSerial>.
- [56] X. Pérez. *the-things-stack-docker: The Things Stack LoRaWAN Network Server on Docker*. GitHub repository, 2025. [Online]. Available: <https://github.com/xoseperez/the-things-stack-docker>. Accessed: Sep. 16, 2025.
- [57] X. Pérez. *basicstation-docker: Docker deployment for Basics™ Station Packet Forward*. GitHub repository, 2025. [Online]. Available: <https://github.com/xoseperez/basicstation-docker>. Accessed: Sep. 16, 2025.



## Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

---

βλ.	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
κ.ά.	και άλλα
LPWAN	Low-Power Wide-Area Network
LoRa	Long Range (φυσικό επίπεδο Semtech)
LoRaWAN	LoRa Wide Area Network (πρωτόκολλο MAC/δικτύου)
TTS	The Things Stack (LoRaWAN Network Server)
LNS	LoRaWAN Network Server
ADR	Adaptive Data Rate
SF	Spreading Factor
BW	Bandwidth
CSS	Chirp Spread Spectrum
MAC	Medium Access Control
MIC	Message Integrity Code
OTAA	Over-The-Air Activation
EUI	Extended Unique Identifier
DevEUI	Device EUI
AppEUI/JoinEUI	Application/Join EUI
DevAddr	Device Address
DR	Data Rate
RX1/RX2	Παράθυρα λήψης downlink 1/2
EU868	Ζώνη συχνοτήτων EU863-870
SPI	Serial Peripheral Interface
GPIO	General-Purpose I/O
REST	Representational State Transfer
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
DB	Database
PF	Power Factor
kWh	κιλοθατώρα
V/A/W/Hz	Volt / Ampere / Watt / Hertz
CLI	Command-Line Interface

TLS	Transport Layer Security
SSH	Secure Shell
DNS	Domain Name System
RPi	Raspberry Pi
PZEM	Peacefair ZEM (μετρητής ενέργειας)