Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Ολοκληρωμένη λύση παρακολούθησης γεωργικών εκτάσεων με ανάπτυξη πρωτότυπου αισθητήρα LoRaWan και χρήση υπηρεσιών νέφους.

Development of a prototype LoRaWan sensor for complete agricultural monitoring solution with use of cloud services.

Σεβεντεκίδης Δημήτριος

Επιβλέποντες Καθηγητές:
Αγγελίδης Παντελής
Γκάλφας Νικόλαος

Κοζάνη, Ιούλιος 2021

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Περίληψη

Η ολοένα και αυξανόμενη ζήτηση γεωργικών προϊόντων σε συνδυασμό με την προσπάθεια βελτίωσης της ποιότητας και του κόστους παραγωγής έχει οδηγήσει την ανθρωπότητα στην επιτακτική ανάγκη εύρεσης σύγχρονων λύσεων. Η εφαρμογή της έξυπνης γεωργίας και της γεωργίας ακριβείας φαίνεται πως μπορούν να δώσουν λύσεις. Η έξυπνη γεωργία ουσιαστικά είναι μια ευρύτερη έννοια κατά την οποία εφαρμόζονται διάφορες τεχνολογίες παρέχοντας κατάλληλες πληροφορίες ικανές να αυξήσουν την γεωργική παραγωγή. Συστήματα έξυπνης γεωργίας μπορούν είτε να συνδυαστούν με συμβατικές μεθόδους γεωργίας είτε να υλοποιηθούν εξ ολοκλήρου από την αρχή αυτοματοποιώντας πολλές διεργασίες για μέγιστη αποδοτικότητα. Για την υλοποίηση τέτοιων συστημάτων συνήθως δημιουργούνται δίκτυα διασυνδεμένων συσκευών, τέτοιες συσκευές είναι αυτές της λήψης μετρήσεων αλλά και των αυτοματισμών. Η δυνατότητα σύνδεσης των συσκευών μεταξύ τους ή με το διαδίκτυο ονομάζεται διαδίκτυο των πραγμάτων (Internet Of Things – IoT). Επίσης, μπορούν να εφαρμοστούν και άλλες τεχνολογίες όπως η χρήση δορυφορικών συστημάτων και αυτόνομων οχημάτων.

Στην παρούσα διπλωματική υλοποιήθηκε ένα ολοκληρωμένο σύστημα παρακολούθησης γεωργικών εκτάσεων με τη χρήση πρωτότυπου αισθητήρα και υπηρεσιών νέφους. Πιο συγκεκριμένα, αρχικά αναπτύχθηκε ενεργειακά αυτόνομος αισθητήριος κόμβος αποστολής μετεωρολογικών δεδομένων με τη χρήση ασύρματου δικτύου χαμηλής ενέργειας ευρείας περιοχής LoRa, τον αισθητήριο κόμβο αποτέλεσαν ο μικροελεγκτής SODAQ ExpLoRer, αισθητήρες ατμόσφαιρας - εδάφους και ηλιακό κύκλωμα τροφοδοσίας. Ο προγραμματισμός του αισθητήριου κόμβου έγινε με τη χρήση του λογισμικού Arduino IDE και η δρομολόγηση των δεδομένων μέσω της πλατφόρμας The Things Network (TTN). Έπειτα, προγραμματίστηκε μια συσκευή Raspberry Pi για χρήση ως εξυπηρετητή. Για της ανάγκες αποθήκευσης και οπτικοποίησης των δεδομένων εγκαταστάθηκαν στον εξυπηρετητή η βάση δεδομένων χρονοσειρών InfluxDB, τα λογισμικό Grafana και Telegraf, επίσης έγινε χρήση υπηρεσιών νέφους (Cloud) για εφεδρική αποθήκευση. Επίσης, αναπτύχθηκε κατάλληλο πρόγραμμα για την αποστολή στατιστικών δεδομένων και ειδοποιήσεων στον τελικό χρήστη με τη χρήση δωρεάν εφαρμογής νέφους. Τέλος, πραγματοποιήθηκε συσχέτιση των δεδομένων του αισθητήριου κόμβου με αυτά πηγών ανοιχτών δεδομένων.

Σκοπός του συγκεκριμένου συστήματος είναι η χρήση του σε γεωργικές εκτάσεις όπου η ειδοποίηση της μεταβολής και η παρακολούθηση της τάσης των καιρικών δεδομένων μπορούν να οδηγήσουν στην έγκαιρη παρέμβαση του ανθρώπου. Ένας ακόμη στόχος της συγκεκριμένης λύσης είναι να μπορεί εύκολα να επεκταθεί και να ενσωματωθεί σε άλλες πλατφόρμες αλλά και σε συστήματα αυτοματισμών.

Λέξεις κλειδιά: έξυπνη γεωργία, γεωργία ακριβείας, δίκτυα, διαδίκτυο των πραγμάτων (IoT), αισθητήριος κόμβος, υπηρεσίες νέφους (cloud), δίκτυα χαμηλής ενέργειας ευρείας περιοχής, LoRa, SODAQ ExpLoRer, αισθητήρες, Arduino IDE, The Things Network, εξυπηρετητής, Raspberry Pi, βάση δεδομένων χρονοσειρών, InfluxDB, Grafana, Telegraf, Telegram

Abstract

The growing demand for agricultural products combined with the effort to improve the quality and cost of production has led humanity to the urgent need to find modern solutions. The application of smart farming and precision farming seems to be able to provide solutions. Smart farming is essentially a wide concept in which various technologies are applied to provide appropriate information capable of increasing agricultural production. Smart farming systems can either be combined with conventional farming methods or implemented entirely from the beginning by automating many processes for maximum efficiency. For the implementation of such systems, device networks are usually being installed, such devices are those of taking measurements but also of automation. The ability of devices to connect to each other or to the Internet is called the Internet of Things (IoT). Other technologies such as the use of satellite systems and autonomous vehicles can also be applied.

In this diploma, an integrated monitoring system of agricultural areas was implemented using a model sensor and cloud services. More specifically, an energy-independent meteorological data transmission node sensor was initially developed using the wide area low power wireless network (LPWAN) LoRa. The node sensor was programmed using Arduino IDE software and the data was routed through The Things Network (TTN) platform. Next, a Raspberry Pi device was programmed to be used as a micro-server. For the needs of data storage and visualization, the InfluxDB time series database, Grafana and Telegraf software were installed on the server, and cloud services were also used for backup storage. Moreover, an appropriate program was developed to send statistical data and alerts to the end user using a free cloud application. Finally, the node sensor data was correlated with API data.

The purpose of this system is to be used in agricultural areas where alerting for change and monitoring the trend of weather data can lead to timely human intervention. Another goal of this solution is to be able to easily expand and integrate into other platforms and automation systems.

Keywords: smart farming, precision farming, networks, internet of things (IoT), node sensor, cloud services, low power wide area networks (LPWAN), LoRa, SODAQ ExpLoRer, sensors, Arduino IDE, The Things Network (TTN), server, Raspberry Pi, time series database, InfluxDB, Grafana, Telegraf, Telegram

Ευχαριστίες

Με την αποπεράτωση της διπλωματικής μου εργασίας και την σηματοδότηση της ολοκλήρωσης των σπουδών μου στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, αρχικά θα ήθελα να ευχαριστήσω όλους τους καθηγητές του τμήματος. Ιδιαίτερα, θα ήθελα να ευχαριστήσω τον Καθηγητή του τμήματος κ. Αγγελίδη Παντελή που μου έδωσε την ευκαιρία να ασχοληθώ και να μάθω ενδιαφέροντα πράγματα από το συγκεκριμένο θέμα ενώ κάθε στιγμή ήταν διαθέσιμος να με βοηθήσει. Επίσης θα ήθελα να ευχαριστήσω θερμά τον κ. Γκάλφα Νίκο που ήταν πάντα διαθέσιμος να μου υποδείξει τον τρόπο σε όποιο τεχνικό αδιέξοδο έφτανα, καθώς και την κ. Καραμήτσου Θωμαή για τον πολύτιμο χρόνο που μου διέθεσε, της οποίας η καθοδήγηση ήταν καθοριστική στην πορεία και την ολοκλήρωση της διπλωματικής εργασίας. Ακόμα, θα ήθελα να ευχαριστήσω τους φίλους και τους συμφοιτητές μου για την υποστήριξή τους τα τελευταία χρόνια. Τέλος, πιο πολύ από όλου θα ήθελα να ευχαριστήσω την οικογένειά μου, που από την πρώτη στιγμή με εμπιστεύτηκαν και με στήριξαν.

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο “Ολοκληρωμένη λύση παρακολούθησης γεωργικών εκτάσεων με ανάπτυξη πρωτότυπου αισθητήρα LoRaWan και χρήση υπηρεσιών νέφους” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

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. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Εισαγωγή

Η συγκεκριμένη διπλωματική εργασία πραγματεύεται την υλοποίηση ολοκληρωμένου συστήματος παρακολούθησης γεωργικών εκτάσεων στα πλαίσια λύσεων έξυπνης γεωργίας. Περιλαμβάνει την υλοποίηση πρωτότυπου αισθητήριου κόμβου βασισμένο στο δίκτυο χαμηλής ενέργειας LoRa, ο οποίος σε συνδυασμό με την κατάλληλη υλικολογισμική υποδομή αποθήκευσης και οπτικοποίησης, παρέχει χρήσιμες πληροφορίες για την πορεία μιας γεωργικής καλλιέργειας στον τελικό χρήστη.

Έξυπνη γεωργία - τρέχουσα κατάσταση και στόχος

Σύμφωνα με μελέτες [1] του Διεθνούς Οργανισμού Γεωργίας και Τροφίμων τα επόμενα χρόνια προβλέπεται σημαντική αύξηση στις ανάγκες παραγωγής γεωργικών προϊόντων. Αυτή η ολοένα και αυξανόμενη ανάγκη για επέκταση της παραγωγής δύναται να αντιμετωπιστεί μόνο επενδύοντας στη γνώση και σε νέες μεθόδους. Η σύγχρονη ιδέα της έξυπνης γεωργίας έρχεται να δώσει λύσεις σε αυτά τα προβλήματα με τα οποία έρχονται αντιμέτωπες οι σύγχρονες καλλιέργειες· συνδυάζοντας μεθόδους συλλογής, επεξεργασίας και αποθήκευσης δεδομένων με τελικό στόχο την κατάλληλη αξιοποίησή τους.

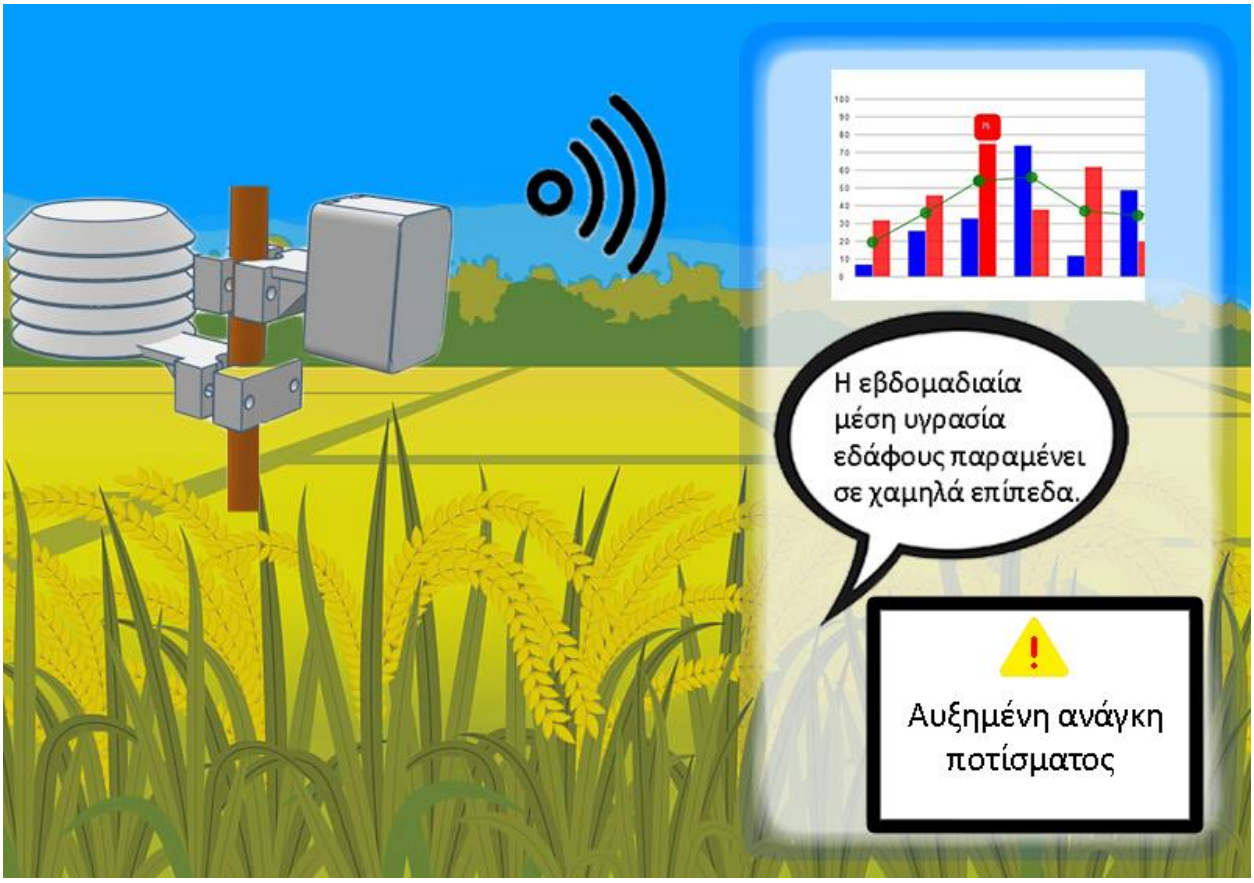
Η έξυπνη γεωργία περιλαμβάνει ένα σύνολο τεχνολογιών και μεθόδων με στόχο τη βελτίωση των συνθηκών, της ποιότητας αλλά και του κόστους παραγωγής. Πολλές φορές συγχέεται με την έννοια της γεωργίας ακριβείας, παρόλα αυτά η γεωργία ακριβείας εστιάζει πιο πολύ στη χρήση δορυφορικών και εναέριων τεχνολογιών οι οποίες βεβαίως συμπεριλαμβάνονται στην ευρύτερη έννοια της έξυπνης γεωργίας. Η έξυπνη γεωργία δεν αποτελεί κάτι άλλο παρά εξέλιξη της συμβατικής γεωργίας με τη συμβολή της τεχνολογίας. Βασίζεται στη συλλογή, επεξεργασία και αποθήκευση δεδομένων, στη χρήση δορυφορικών και εναέριων τεχνολογιών καθώς και στην υλοποίηση αυτοματισμών και ρομποτικών μηχανημάτων.

Όσον αφορά τη διαδικασία της συλλογής, επεξεργασίας και αποθήκευσης δεδομένων, ουσιαστικά πρόκειται για τη χρήση ειδικά σχεδιασμένων αισθητήρων που στοχεύουν στη λήψη δεδομένων μετρήσεων από μια αγροτική έκταση. Τέτοια δεδομένα μπορούν να αποτελούν οι καιρικές συνθήκες που επικρατούν. Η καταγραφή των καιρικών συνθηκών που επικρατούν σε μια συγκεκριμένη αγροτική έκταση κρίνεται ιδιαίτερα σημαντική καθώς μπορεί να συμβάλει στην καλύτερη διαχείριση και αντιμετώπιση επερχόμενων κινδύνων. Επίσης παρακολουθώντας την τάση του καιρού ο γεωργός είναι σε θέση να κάνει πιο σωστή τη διαδικασία λίπανσης και την κατανομή των αρδευτικών πόρων. Σε τέτοια σενάρια χρήσης μετεωρολογικών αισθητήρων, δημιουργούνται δίκτυα αποτελούμενα από αισθητήριους κόμβους ικανά να καλύψουν μεγάλες αγροτικές εκτάσεις που η διαφορά των καιρικών συνθηκών από κόμβο σε κόμβο είναι σημαντική. Έπειτα τα δεδομένα αποθηκεύονται και επεξεργάζονται με τη χρήση μετεωρολογικών μοντέλων δίνοντας στον τελικό χρήστη τις απαραίτητες πληροφορίες.

Κατά την υλοποίηση τέτοιων συστημάτων παρακολούθησης καιρικών συνθηκών με τη χρήση αισθητήρων πρέπει να αντιμετωπιστούν τα προβλήματα κάλυψης δικτύου και παροχής ηλεκτρικής ενέργειας. Καθώς οι γεωργικές εκτάσεις που χρίζουν καιρικής παρακολούθησης τις περισσότερες φορές

είναι μακριά από αστικές περιοχές και η απευθείας αποστολή δεδομένων μέσω διαδικτύου είναι αδύνατη, τα δίκτυα LPWAN έρχονται να δώσουν λύση. Επίσης, τις ανάγκες τροφοδότησης με ηλεκτρική ενέργεια συνήθως αναλαμβάνουν ηλιακά πάνελ μαζί με επαναφορτιζόμενες μπαταρίες.

Ο αρχικός στόχος στην παρούσα διπλωματική εργασία είναι να υλοποιηθεί ολοκληρωμένη λύση παρακολούθησης γεωργικών εκτάσεων με χρήση πρωτότυπου αισθητήρα και υπηρεσιών νέφους. Ακολουθεί αναλυτική περιγραφή της ανάπτυξης του πρωτότυπου αισθητήριου κόμβου, αποτελούμενο από αισθητήρες ατμοσφαίρας και εδάφους, επίσης αναλύεται ο τρόπος χρήσης κατάλληλου κυκλώματος ηλιακής ενέργειας. Για τις ανάγκες αποθήκευσης των δεδομένων θα γίνει χρήση βάσης δεδομένων χρονοσειρών. Τέλος, το ιδανικό αποτέλεσμα είναι να παρέχονται στον τελικό χρήστη χρήσιμες πληροφορίες σχετικές με τις καιρικές συνθήκες που επικρατούν σε μια γεωργική έκταση αποσκοπώντας στη βελτίωση της ποιότητας και μείωση του κόστους παραγωγής. Επίσης, είναι επιθυμητό, η υποδομή του συστήματος να μπορεί να επεκταθεί και να συνδυαστεί με άλλα συστήματα έξυπνης γεωργίας και αυτοματισμών. Στα προβλήματα που αναμένεται να δώσει λύσεις η συγκεκριμένη διπλωματική εργασία αφορούν στην κάλυψη μεγάλων αποστάσεων, την ενεργειακή αυτονομία του αισθητήριου κόμβου, καθώς και τη μη εξάρτηση σε μηνιαία κόστη για την αποστολή δεδομένων, όπως αυτά της χρήσης κυψελοειδών δικτύων για την αποστολή μηνυμάτων sms.



Εικόνα 1: Αρχικός στόχος

Ενδεικτικά έργα και εφαρμογές

Η επιστημονική κοινότητα έχει αντιληφθεί την επιτακτική ανάγκη εύρεσης σύγχρονων λύσεων για την κάλυψη των αυξανόμενων αναγκών για βελτίωση της γεωργικής παραγωγής. Όσο περνάει ο καιρός ολοένα και περισσότερα συστήματα έξυπνης γεωργίας ενσωματώνονται στη διαδικασία της συμβατικής γεωργίας. Ακολουθούν κάποια ενδεικτικά ερευνητικά έργα και εφαρμογές που στοχεύουν στη βελτίωση των συστημάτων έξυπνης γεωργίας.

Πανεπιστήμιο Δυτικής Μακεδονίας

Το πανεπιστήμιο Δυτικής Μακεδονίας και συγκεκριμένα το τμήμα των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών διαδραματίζει σημαντικό ρόλο στην υλοποίηση και τη βελτιστοποίηση συστημάτων έξυπνης γεωργίας συμμετέχοντας σε σημαντικά ερευνητικά έργα.

Ερευνητικό έργο ΑΥΓΕΙΑΣ

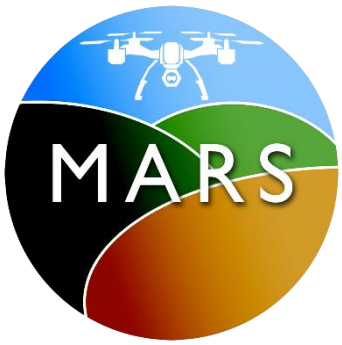
Το ερευνητικό έργο ΑΥΓΕΙΑΣ [2] στο οποίο συμμετέχει και το πανεπιστήμιο Δυτικής Μακεδονίας έχει ως στόχο τη δημιουργία ενός ολοκληρωμένου έξυπνου συστήματος βελτιστοποίησης της χρήσης ανακυκλωμένου νερού προερχόμενο από βιολογικό καθαρισμό για αρδευτική χρήση στη γεωργία. Στοχεύει στην υλοποίηση ενός δικτύου LPWAN στο οποίο θα συνδέονται ασύρματοι αισθητήρες IoT και θα αποστέλλονται δεδομένα πραγματικού χρόνου. Έπειτα μια πλατφόρμα συλλογής των δεδομένων σε συνδυασμό με μεθόδους μηχανικής μάθησης και συσχετισμό με ανοιχτά δεδομένα θα είναι σε θέση να λαμβάνει αποφάσεις και να ενεργοποιεί κατάλληλους αυτοματισμούς για βέλτιστη διαχείριση του ανακυκλώσιμου νερού. Η πλατφόρμα θα περιλαμβάνει σύστημα πρόβλεψης της πορείας της καλλιέργειας ενός παραγωγού, καθώς και έξυπνο σύστημα τιμολόγησης τόσο για αύξηση των κερδών της διαχειριστικής αρχής όσο και για την μείωση των εξόδων του παραγωγού.

Εικόνα 2: Λογότυπο ερευνητικού έρου Αυγείας

Ερευνητικό έργο MARS

Το ερευνητικό έργο sMart fArming with dRoneS (MARS) [3] του οποίου την υλοποίηση έχει αναλάβει το τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας στοχεύει στην ενσωμάτωση τεχνολογιών πληροφορικής και επικοινωνιών στη διαδικασία γεωργικής παραγωγής. Πιο συγκεκριμένα έχει ως στόχο τη χρήση μη επανδρωμένων αεροσκαφών για την εποπτεία γεωργικών εκτάσεων με αναμενόμενο αποτέλεσμα την έγκαιρη ανίχνευση κινδύνων μέσα από ένα ολοκληρωμένο σύστημα ελέγχου. Ο τελικός σκοπός του έργου

είναι η αύξηση της ποιότητας και η μείωση του κόστους παραγωγής χρησιμοποιώντας λιγότερα φυτοπροστατευτικά προϊόντα και αντιμετωπίζοντας τις απειλές με έξυπνες μεθόδους μηχανικής μάθησης.



Εικόνα 3: Λογότυπο έργου MARS

JOHN DEERE

Η αμερικανική εταιρία John Deere [4] η οποία δραστηριοποιείται στην κατασκευή εξειδικευμένων γεωργικών μηχανημάτων και εργαλείων έχει στραφεί στην έξυπνη γεωργία όταν ήδη από το 2002 εισήγαγε στη αγορά το AutoTrac, ένα αυτόνομο τρακτέρ ικανό να χαράσσει την πορεία του με τη χρήση συστημάτων GPS. Έκτοτε η John Deere έχει κατασκευάσει σύγχρονα γεωργικά μηχανήματα με εξαιρετική ακρίβεια ενσωματώνοντας εξιδεικευμένους αισθητήρες και εναέρια μέσα με κατάλληλα λογισμικά μηχανικής μάθησης και ανάλυσης δεδομένων. Η John Deere σήμερα προσφέρει ολοκληρωμένες λύσεις έξυπνης γεωργίας, από την συλλογή μετεωρολογικών δεδομένων μέσω IoT αισθητήρων έως τον ψεκασμό καλλιεργειών με αυτόνομα μη επανδρωμένα αεροσκάφη.

Εικόνα 4: Εφαρμογή έξυπνης γεωργίας από την john deere

Δομή διπλωματικής εργασίας.

Η εργασία δομείται σε τέσσερα κεφάλαια τα οποία περιγράφουν όλη τη διαδικασία υλοποίησης του αισθητήριου κόμβου καθώς και την χρήση λογισμικού προκειμένου να γίνει σωστή αξιοποίηση των δεδομένων που αντλούνται.

Στο πρώτο κεφάλαιο γίνεται εκτενής περιγραφή όλων των εννοιών και των τεχνολογιών που συνέβαλαν στην υλοποίηση της συγκεκριμένης διπλωματικής εργασίας. Πιο συγκεκριμένα, αναλύεται το ίντερνετ των πραγμάτων (IoT) που σε συνδυασμό με τα LPWAN δίκτυα, συγκεκριμένα το LoRaWAN και την τεχνολογία MQTT θεμελιώνουν την αποστολή των δεδομένων. Έπειτα ακολουθεί η περιγραφή του προγραμματιστικού υποβάθρου, το οποίο περιλαμβάνει τις γλώσσες και τα εργαλεία προγραμματισμού. Τέλος ακολουθεί εξήγηση των εργαλείων και λογισμικών των οποίων η συνεισφορά για την αποθήκευση, την οπτικοποίηση και την αποστολή των δεδομένων ήταν απαραίτητη.

Στο δεύτερο κεφάλαιο αναλύεται το υλικό μέρος που χρησιμοποιήθηκε για την υλοποίηση του κόμβου LoRa , καθώς επίσης περιγράφεται βήμα προς βήμα όλη η διαδικασία προγραμματισμού του αισθητήριου κόμβου μέσα από το προγραμματιστικό περιβάλλον Arduino IDE. Περιγράφεται επίσης η διαδικασία σύνδεσης και αποστολής των δεδομένων του κόμβου στο The Things Network .

Στο τρίτο κεφάλαιο αναλύεται η υποδομή λογισμικού του συστήματος που αναπτύχθηκε με σκοπό την αξιοποίηση των δεδομένων του αισθητήριου κόμβου. Αυτό το σύστημα περιλαμβάνει υλοποίηση βάσης δεδομένων σε InfluxDB καθώς και ανάπτυξη κατάλληλης υπηρεσίας αλληλεπίδρασης με τον τελικό χρήστη μέσω οπτικοποίησης και αποστολής μηνυμάτων σχετικά με τα δεδομένα του αισθητήριου κόμβου. Επίσης εξηγείται η διαδικασία προώθησης των δεδομένων από το The Things Network στη βάση δεδομένων μέσω της υπηρεσίας Telegraf. Τέλος παρουσιάζεται η υλοποίηση του micro-server που υλοποιήθηκε σε Raspberry Pi , για την εκτέλεση όλων των απαραίτητων διεργασιών.

Στο τέταρτο κεφάλαιο της περιγράφεται η εφαρμογή της ολοκληρωμένης λύσης σε πραγματικές συνθήκες γεωργίας. Το συγκεκριμένο κεφάλαιο κρίνεται ιδιαίτερα σημαντικό καθώς έδωσε την ευκαιρία εύρεσης και επίλυσης σφαλμάτων με τελικό στόχο τη βελτιστοποίηση του συστήματος.

Στο πέμπτο και τελευταίο κεφάλαιο της διπλωματικής εργασίας πραγματοποιήθηκε αποθήκευση δεδομένων από προγραμματιστικές διεπαφές που διαθέτουν μετεωρολογικές πλατφόρμες, με στόχο τη συσχέτισή τους με αυτά του αισθητήριου κόμβου. Στόχο της συσχέτισης αποτέλεσε η ανάγκη αξιολόγησης των δεδομένων του πρωτότυπου αισθητήρα αλλά και η δυνατότητα εναλλακτικής λήψης δεδομένων από τον τελικό χρήστη.

Τέλος ακολουθεί ο επίλογος όπου γίνεται σύνοψη της διπλωματικής εργασίας και αναφορά στα προβλήματα που αντιμετωπίστηκαν σε όλη τη διάρκεια εκπόνησής της, καθώς και στις δυνατότητες μελλοντικής επέκτασης.

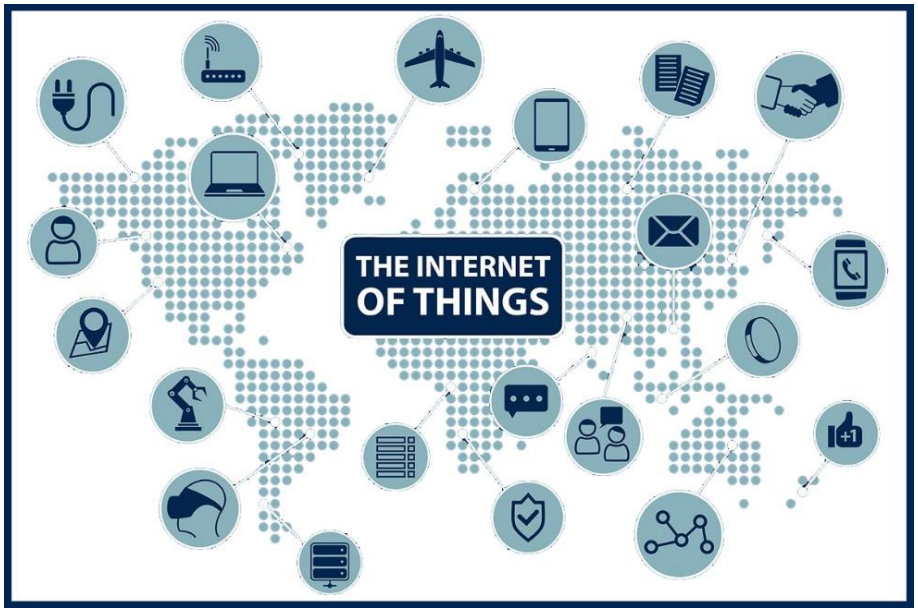
Κεφάλαιο 1ο - Θεωρητικό Υπόβαθρο

1.1 Βασικές Έννοιες

1.1.1 Internet Of Things

Ο όρος Internet Of Things-Διαδίκτυο των πραγμάτων (IoT) [5] επινοήθηκε από τον ιδρυτή του Auto-ID Center Kevin Ashton στις αρχές του 2000. Την πρώτη εφαρμογή IoT αποτέλεσε η δικτύωση ετικετών RFID που υλοποιήθηκε στα πλαίσια του Auto-ID Center. Το Internet Of Things αποτελείτε από δυο βασικά στοιχεία , τη δυνατότητα σύνδεσης στο διαδίκτυο ή γενικότερα σε κάποιο δίκτυο «Internet» και τις διάφορες ηλεκτρονικές συσκευές που διαθέτουν μοναδικό αναγνωριστικό και έχουν τη δυνατότητα σύνδεσης είτε μεταξύ τους , είτε στο εκάστοτε δίκτυο. Τις συσκευές , «Things» δεν αποτελούν συσκευές στα πλαίσια αυστηρών προτύπων . Πολλές συσκευές με διαφορετικές αρχιτεκτονικές και λειτουργίες μπορούν να αποτελούν τα Things ενός δικτύου με το μοναδικό κοινό χαρακτηριστικό να είναι η δυνατότητα διασύνδεσης .

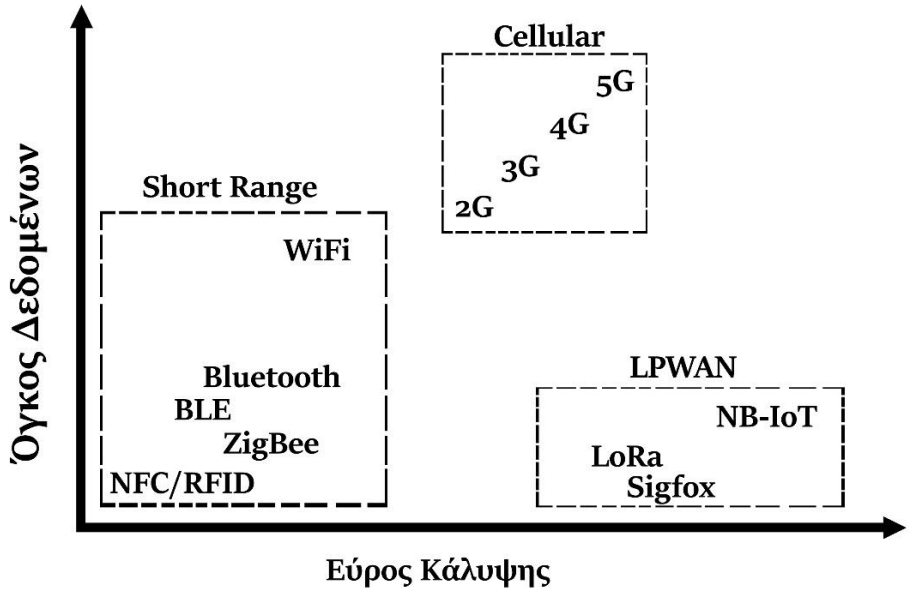
Τα πρώτα δίκτυα IoT εφαρμόστηκαν σε εξειδικευμένες επιχειρήσεις , όπου η κύρια λειτουργία τους ήταν η σύνδεση αισθητήρων M2M (machine to machine) με σκοπό την λήψη μετρήσεων προς όφελος της γραμμής παραγωγής. Πλέον IoT εφαρμογές συναντώνται τόσο σε απλές οικιακές συσκευές όσο και σε πιο ειδικές , που μπορεί να είναι κάποιο δίκτυο αισθητήρων για την καταγραφή της λειτουργίας των συσκευών σε ένα πλοίο μεταφοράς κοντέινερ ή κάποιο δίκτυο αισθητήρων μιας ολοκληρωμένης εφαρμογής γεωργίας ακριβείας.



Εικόνα 5: Το διαδίκτυο των πραγμάτων

Ο ουσιαστικός σκοπός του IoT είναι η δυνατότητα του χρήστη είτε να ελέγχει τις συνδεδεμένες συσκευές είτε να αξιοποιεί κατάλληλα τα δεδομένα που του παρέχουν. Το σύνηθες σενάριο λειτουργίας είναι , συσκευές να συνδέονται σε μια πλατφόρμα και να μοιράζονται τις πιο χρήσιμες πληροφορίες, εδώ προκύπτει και η πρώτη πρόκληση που θα αναλυθεί παρακάτω. Η δυνατότητα διασύνδεσης και αξιοποίησης των δεδομένων όλων αυτών των συσκευών μπορεί να αυτοματοποιήσει χρονοβόρες ή ακόμα και επικίνδυνες διαδικασίες. Το IoT μπορεί να αυξήσει την παραγωγή σε εργοστάσια αλλά και την ποιότητα των υπηρεσιών σε υγειονομικές εφαρμογές.

Πρόκληση αποτελεί η διαχείριση και η ερμηνεία του τεράστιου όγκου των δεδομένων (Big Data) που εξάγουν οι συσκευές που αλληλοεπιδρούν με το δίκτυο σε πραγματικό χρόνο. Κατά την υλοποίηση τέτοιων IoT συστημάτων θα πρέπει να αναπτύσσονται τεχνικές ώστε να λαμβάνονται από τις συσκευές τα δεδομένα που χρειάζονται τη σωστή στιγμή και στο τέλος να αποθηκεύονται μόνο τα απολύτως απαραίτητα δεδομένα . Η βελτιστοποίηση των IoT συστημάτων μπορεί να συμβάλει τόσο στη μείωση ζήτησης της ηλεκτρικής ισχύς όσο και στην αποσυμφόρηση των βάσεων δεδομένων . Τρόπους σύνδεσης των ασύρματων συσκευών αποτελούν οι επικοινωνίες κοντινής απόστασης «Short Range Communications» , οι κυψελοειδείς επικοινωνίες «Cellular communications» και οι επικοινωνίες ευρείας απόστασης χαμηλής ισχύος «LPWAN communications» .



Εικόνα 6: Σύγκριση ασύρματων τεχνολογιών σε σχέση με τον όγκο δεδομένων και το εύρος κάλυψης

Στα πλαίσια του IoT , διασυνδέσεις κοντινών αποστάσεων επιτυγχάνονται συνήθως μέσω των τεχνολογιών NFC/RFID , Zigbee , BLE, Bluetooth και WiFi, ενώ διασυνδέσεις κυψελοειδών επικοινωνιών επιτυγχάνονται μέσω 2G , 3G , 4G και 5G και τέλος διασυνδέσεις ευρείας απόστασης χαμηλής ισχύος επιτυγχάνονται μέσω NB-IoT , Sigfox και **LoRa**.

Ασύρματη Τεχνολογία	**Ασύρματη Επικοινωνία**	**Εύρος**	**Ισχύς Μετάδοσης (Tx-Power)**
Bluetooth	Short Range	~10m	~2.5 mW
WiFi	Short Range	~50m	~80 mW
3G/4G	Cellular	~5000m	~500 mW
LoRa	LPWAN	*Εξαρτάται από τις συνθήκες. • Σε αστική περιοχή: 2km →5km • Σε μη αστική περιοχή: 5km→15km • Σε συνθήκες ανεμπόδιστης οπτικής επαφής: >15km	~20 mW

Πίνακας 1: Χαρακτηριστικά ασύρματων τεχνολογιών

1.1.2 Δίκτυα LPWAN – LoRa

Στην συγκεκριμένη διπλωματική εργασία σημαντικό ρόλο παίζει η επικοινωνία των δικτύων LPWAN (Low Power Wide Area Network) [6] ,πιο συγκεκριμένα η μετάδοση των δεδομένων από τον αισθητήριο κόμβο που υλοποιήθηκε έγινε με LoRa. Τα βασικά χαρακτηριστικά των δικτύων LPWAN είναι η αποστολή μικρών πακέτων δεδομένων σε μεγάλες αποστάσεις και ταυτόχρονα οι ελάχιστες ενεργειακές απαιτήσεις. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο οι πιο διαδεδομένες τεχνολογίες LPWAN σε IoT εφαρμογές είναι οι τεχνολογίες NB-IoT , Sigfox και LoRa.

1.1.2.1 Narrowband Internet of Things

Το Narrowband Internet of Things (NB-IoT) [7] είναι μια τεχνολογία LPWAN που αναπτύχθηκε στα πλαίσια του 3rd Generation Partnership Project (3GPP). Μπορεί να συνυπάρξει με τα δίκτυα GSM και LTE στις συχνότητες των 700MHz , 800MHz και 900MHz, στην πραγματικότητα χρησιμοποιεί ένα υποσύνολο του LTE εύρους ζώνης 200KHz . Το NB-IoT χρησιμοποιεί πολλαπλή πρόσβαση FDMA και με μετάδοση της τάξης των 200 bytes την ημέρα μια συσκευή που βασίζεται σε αυτή την τεχνολογία μπορεί να επιτύχει μέχρι και 10 χρόνια λειτουργείας με μια μπαταρία. Τέλος, το NB-IoT χρησιμοποιείται κυρίως σε IoT εφαρμογές ασφάλειας ,μεταδίδοντας δεδομένα αισθητήρων και όχι μόνο. Το κύριο πλεονέκτημα αυτής της τεχνολογίας έχει να κάνει με την υπεροχή στο low Latency και στο Quality of Service.

1.1.2.2 Sigfox

Η τεχνολογία Sigfox [8] αναπτύχθηκε από την ομώνυμη εταιρία Sigfox στην Τουλούζη της Γαλλίας το 2010. Λειτουργεί και διατίθεται αποκλειστικά από το δίκτυο IoT που έχει στήσει η ίδια η εταιρία σε πολλές χώρες και συνεχώς επεκτείνεται μέσω εταιριών-συνεργατών στον τομέα των δικτύων. Οι τελικές συσκευές-κόμβοι συνδέονται στους σταθμούς βάσης (gateways) με διαμόρφωση Binary Phase Shift Keying (BPSK) σε Ultra-Narrow band στα 100Hz που συναντάτε κάτω από την συχνότητα των GHZ ISM (sub-GHZ ISM). Η Sigfox δουλεύει σε ελεύθερες ISM ζώνες στα 868MHz στην Ευρώπη , στα 915MHz στη Βόρειο Αμερική και στα 433MHz στην Ασία , χρησιμοποιώντας την εξαιρετικά στενή ζώνη που αναφέρθηκε προηγουμένως το Sigfox χρησιμοποιεί πολύ αποτελεσματικά το εύρος ζώνης αντιμετωπίζοντας τα επίπεδα θορύβου και κρατώντας τις ενεργειακές ανάγκες χαμηλά. Για την επικοινωνία κόμβου – σταθμού βάσης και την ανταλλαγή μηνυμάτων απαιτείτε το αρχικό μήνυμα ζεύξης. Τα μηνύματα που μπορεί να στείλει ο κάθε κόμβος (downlink) καθημερινά είναι 140 και ο όγκος των δεδομένων που μπορεί να υποστηρίξει ένα downlink μήνυμα , περιορίζετε στα 12 byte , ωστόσο ο περιορισμός των uplink μηνυμάτων περιορίζει τον κόμβο στο να δέχεται 4 μηνύματα ημερησίως. Το βασικό πλεονέκτημα του Sigfox έχει να κάνει με την χαμηλή ενεργειακή κατανάλωση και χρησιμοποιείτε κυρίως σε IoT εφαρμογές γενικών μετρήσεων , smart cities και παρακολούθησης κυκλοφορίας.

1.1.2.3 LoRa – LoRaWAN

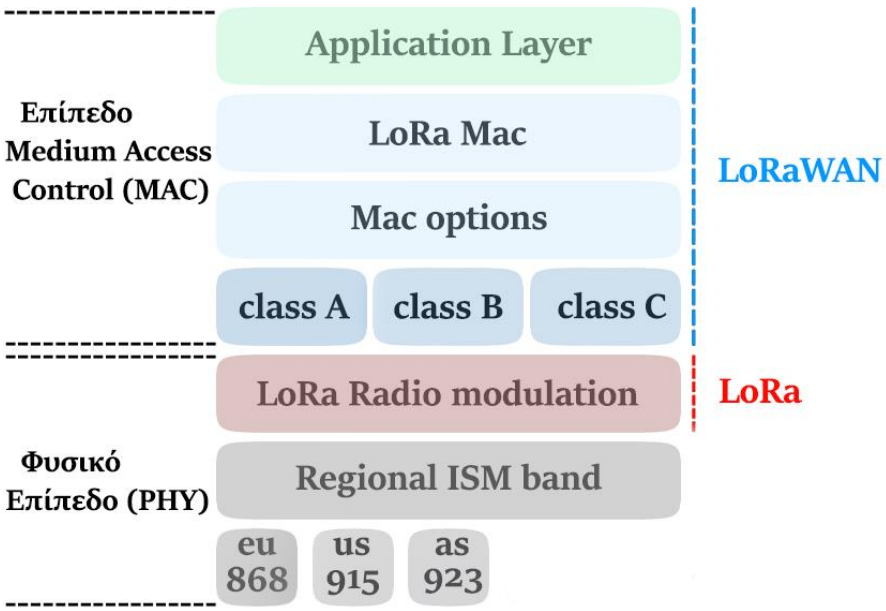
Η τεχνολογία LoRa (**Lo** ng **Ra** nge) [9] αναπτύχθηκε αρχικά από την start-up Cycleo το 2009 στη Γαλλία με την κατασκευή κατάλληλων microchips, έπειτα από τρία χρόνια η Semtech (ΗΠΑ) εξαγόρασε την μέχρι τότε τεχνολογία και την εξελίσσει μέχρι σήμερα με την υποστήριξη της κοινοπραξίας LoRa Alliance.

Το LoRa αποτελεί και αυτό ένα δίκτυο LPWAN , φυσικού επιπέδου που διαμορφώνεται στη ζώνη sub-GHZ ISM χρησιμοποιώντας μια αποκλειστική τεχνική εξάπλωσης του φάσματος. Χρησιμοποιεί τις

συχνότητες των 868MHz στην Ευρώπη , 915MHz στην Αυστραλία και τη Βόρεια Αμερική , 865MHz - 867MHz στην Ινδία και 923MHz στην Ασία. Η αμφίδρομη επικοινωνία επιτυγχάνεται μέσω διαμόρφωσης chirp spread spectrum (CSS) η οποία διαδίδει ένα σήμα narrow-band, με παρόμοιο τρόπο των τεχνολογιών NB-IoT και Sigfox. Το σήμα που διαδίδεται έχει χαμηλά επίπεδα θορύβου, γεγονός που το καθιστά ανθεκτικό στις παρεμβολές και τον εντοπισμό.

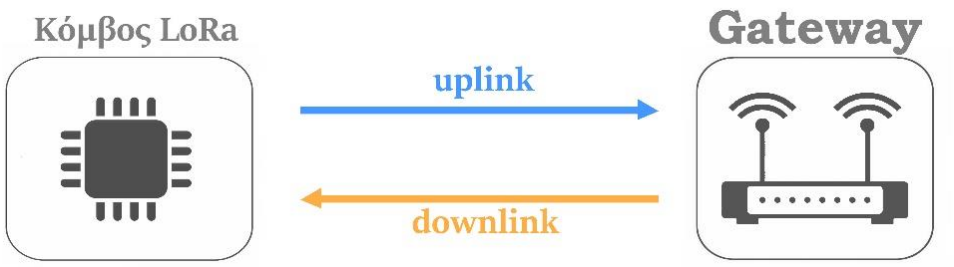
Το LoRa χρησιμοποιεί έξι παράγοντες διασποράς (Spreading Factors -SF) από 7 έως 12 ώστε να επιτύχει βέλτιστη προσαρμογή του ρυθμού δεδομένων με το εύρος της ανταλλαγής. Υψηλός ρυθμός διασποράς σημαίνει μεγαλύτερο εύρος διάδοσης με χαμηλό ρυθμό δεδομένων (data rate), και το αντίστροφο. Ο ρυθμός δεδομένων κυμαίνεται μεταξύ 300 bps και 27 kbps , με διαμόρφωση FSK μπορεί να επιτευχθεί ρυθμός έως 50 kbps. Το μέγιστο μήκος ενός μηνύματος που μπορεί να σταλεί από έναν κόμβο (payload message) ορίζεται στα 243 byte. Ένας κόμβος LoRa (end node – τελική συσκευή) είναι μια συσκευή που διαθέτει κατάλληλο radio module με κεραία και μικροεπεξεργαστή για την επεξεργασία των δεδομένων. Τους σταθμούς βάσεις (gateways) αποτελούν συσκευές που διαθέτουν και αυτές κατάλληλο radio module με κεραία και μικροεπεξεργαστή , παράλληλα διαθέτουν και συνδεσιμότητα με το διαδίκτυο είτε μέσω Ethernet ή WiFi , είτε μέσω LTE.

Η LoRa Alliance το 2015 τυποποίησε το πρωτόκολλο επικοινωνίας LoRaWAN που βασίζεται σε επικοινωνίες με βάση την τεχνολογία LoRa. Κάθε μήνυμα που μεταδίδει μια συσκευή χρησιμοποιώντας το παραπάνω πρωτόκολλο λαμβάνεται από όλους τους σταθμούς βάσης (gateways) που προσφέρουν κάλυψη.



Εικόνα 7: Στοίβα πρωτοκόλλου LoRa

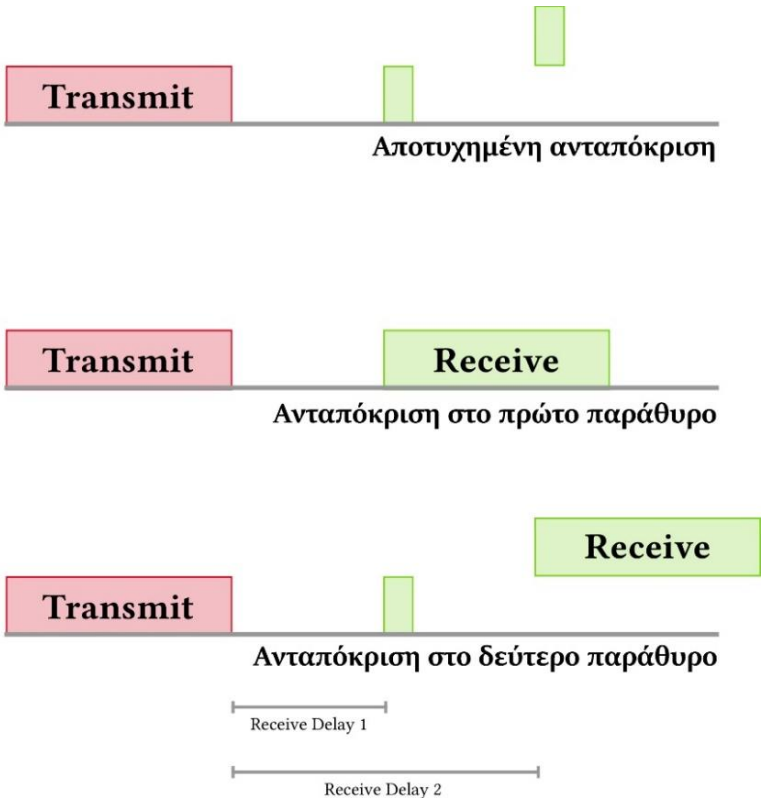
Το LoRaWAN εκμεταλλεύεται αυτή την μη-αναγκαία λήψη μηνύματος από επιπλέον σταθμούς βάσης , ώστε να μεγιστοποιεί την πιθανότητα λήψης του εκάστοτε μηνύματος. Ο διακομιστής του δικτύου (network server) είναι υπεύθυνος για το φιλτράρισμα των πιθανών διπλότυπων μηνυμάτων καθώς επίσης για την ασφάλεια και την ομαλή ροή των δεδομένων προς τρίτες εφαρμογές. Πολλά μηνύματα από τον ίδιο κόμβο που καταφθάνουν σε διαφορετικούς σταθμούς βάσης μπορούν να χρησιμοποιηθούν για τον ακριβή γεωεντοπισμό του κόμβου με χρήση της τεχνικής TDOA , η οποία έχει τη δυνατότητα με την χρήση ενός επιβεβαιωμένου μηνύματος σε τρεις ή περισσοτέρους σταθμούς βάσης να υπολογίσει την ακριβή θέση με βάση την διαφορά στους χρόνους άφιξης του κάθε μηνύματος.



Εικόνα 8: Ανταλλαγή μηνυμάτων κόμβου - Gateway

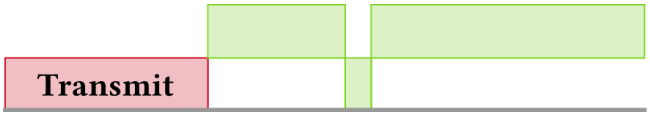
Οι τελικές συσκευές (κόμβοι) στο LoRaWAN κατηγοριοποιούνται σε τρεις κατηγορίες (κλάσεις) ανάλογα με τον τρόπο που διαχειρίζονται τα μηνύματα αποστολής (downlink) και λήψης (uplink).

- Οι συσκευές κλάσης A (class A) , υποστηρίζουν αμφίδρομη επικοινωνία ανάμεσα στις ίδιες και τους σταθμούς βάσης . Ένα uplink μήνυμα μπορεί να αποστέλλεται σε οποιαδήποτε χρονική στιγμή , και ακολουθείται από δυο παράθυρα σε προκαθορισμένες στιγμές . Αν το gateway δεν ανταποκριθεί σε κανένα από τα δυο παράθυρα στους προκαθορισμένους χρόνους , θα πρέπει να περιμένει το επόμενο uplink. Το gateway μπορεί να ανταποκριθεί είτε στο πρώτο, είτε στο δεύτερο παράθυρο ,αλλά όχι και στα δύο.



Εικόνα 9: Αμφίδρομη επικοινωνία συσκευών κλάσης A

- Όσον αφορά τις συσκευές κλάσης Β (class B) , όπως και οι συσκευές κλάσης Α υποστηρίζουν αμφίδρομη επικοινωνία ανάμεσα στις ίδιες και τους σταθμούς βάσης, ανοίγοντας όμως επιπλέον περιοδικά παράθυρα λήψης σε προγραμματισμένες στιγμές. Οι στιγμές καθορίζονται από έναν συγχρονισμένο φάρο που παρέχει το gateway.
- Οι συσκευές κλάσης Γ (class C), υποστηρίζουν και αυτές αμφίδρομη επικοινωνία διατηρώντας τα παράθυρα λήψης ανοιχτά όσο δεν μεταδίδουν κάποιο μήνυμα. Αυτές οι συσκευές καταναλώνουν συνήθως περισσότερη ενέργεια.



Εικόνα 10: Αμφίδρομμη επικοινωνία συσκευών κλάσης Β

Όσον αφορά τη χρήση του LoRa , το συναντάμε κυρίως σε IoT εφαρμογές σαν αυτές που ακολουθούν:

- Έξυπνες Εφαρμογές – Smart Utilities:
  - Επίβλεψη ηλεκτρικής ισχύος.
  - Επίβλεψη στάθμης νερού.
  - Επίβλεψη καυσίμου.

- Εφαρμογές Υγείας και Υγιεινής
  - Επίβλεψη αρτηριακής πίεσης.
  - Επίβλεψη θερμοκρασίας.

- Εφαρμογές Ασφάλειας
  - Έξυπνος φωτισμός.
  - Επίβλεψη επιπέδων ραδιενέργειας.

- Εφαρμογές Αποδοτικότητας
  - Επίβλεψη οχημάτων μέσω GPS.
  - Επίβλεψη Containers.

- Εφαρμογές έξυπνης Κτηνοτροφίας/Γεωργίας
  - Επίβλεψη συνθηκών ζώων σε κτηνοτροφικές μονάδες.
  - Επίβλεψη συνθηκών καλλιέργειας.

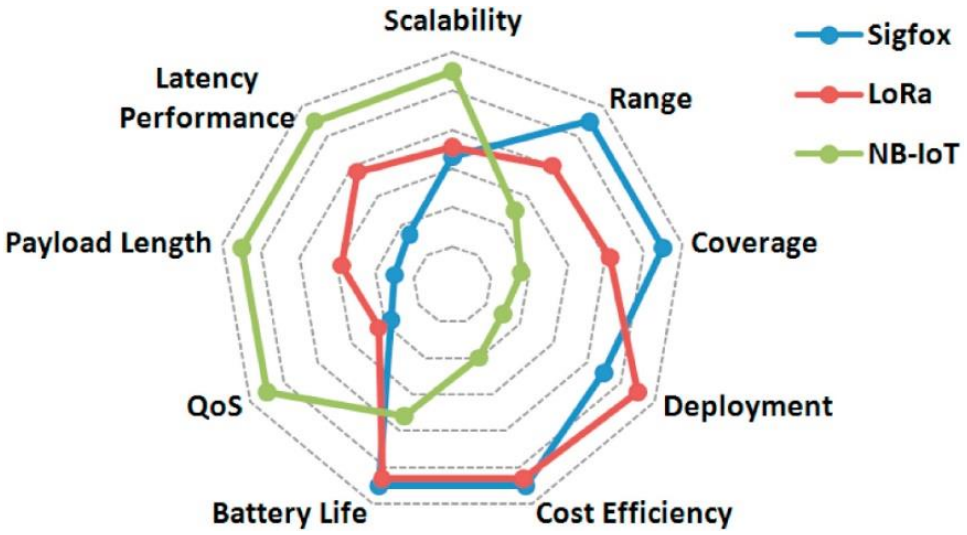
Το βασικό πλεονέκτημα του LoRaWAN έγκειται στο ότι μπορεί να μεταδίδει μηνύματα σε μεγάλες αποστάσεις με πολύ μικρή κατανάλωση ενέργειας καθώς και στο ότι το κόστος μια συσκευής που το υποστηρίζει είναι πολύ μικρό. Τεχνολογία LoRa δεν μπορεί να χρησιμοποιηθεί για αποστολή δεδομένων μεγάλου όγκου , όπως αρχεία φωτογραφιών ή βίντεο παρά μόνο για μικρά πακέτα δεδομένων , όπως δεδομένα IoT αισθητήρων. Η εμβέλεια του ποικίλει ανάλογα με τη γεωμορφολογική τοπολογία του εδάφους και τα πιθανά εμπόδια, αξίζει να σημειωθεί το ρεκόρ του Andreas Spiess ο οποίος πέτυχε ανταλλαγή μηνυμάτων στα 212km με επικοινωνία εδάφους-εδάφους. Επίσης το 2017 κατά τη διάρκεια της ετήσιας εκδήλωσης Koppelting επιτεύχθηκε επικοινωνία 702 χιλιομέτρων μέσω ενός μετεωρολογικού μπαλονιού. Το LoRaWAN είναι ένα πρωτόκολλο υπό ανάπτυξη και η κοινοπραξία LoRa Alliance συνεχίζει να το εξελίσσει και αναμένεται να προστεθούν νέες δυνατότητες σε επόμενες εκδόσεις.

Χαρακτηριστικά	LoRa
Συχνότητα(Frequency)	433MHz – 915MHz
Εύρος ζώνης(Bandwidth)	0.125Mhz , 0.250MHz
Ρυθμός Δεδομένων(Data Rate)	300bps – 50 kbps
Ισχύς εκπομπής(Transmission Power)	14dBm
Εμβέλεια	*έως 15km
Μέγεθος πακέτου	Έως 243bytes
Κρυπτογράφηση	AES 128b

Πίνακας 2: Χαρακτηριστικά LoRa

1.1.2.4 Σύνοψη Ενότητας

Στη συγκεκριμένη ενότητα έγινε αναφορά σε ορισμένα δίκτυα LPWAN που χρησιμοποιούνται κυρίως για την μετάδοση δεδομένων IoT εφαρμογών, επίσης έγινε πιο εκτεταμένη ανάλυση της τεχνολογίας LoRa.



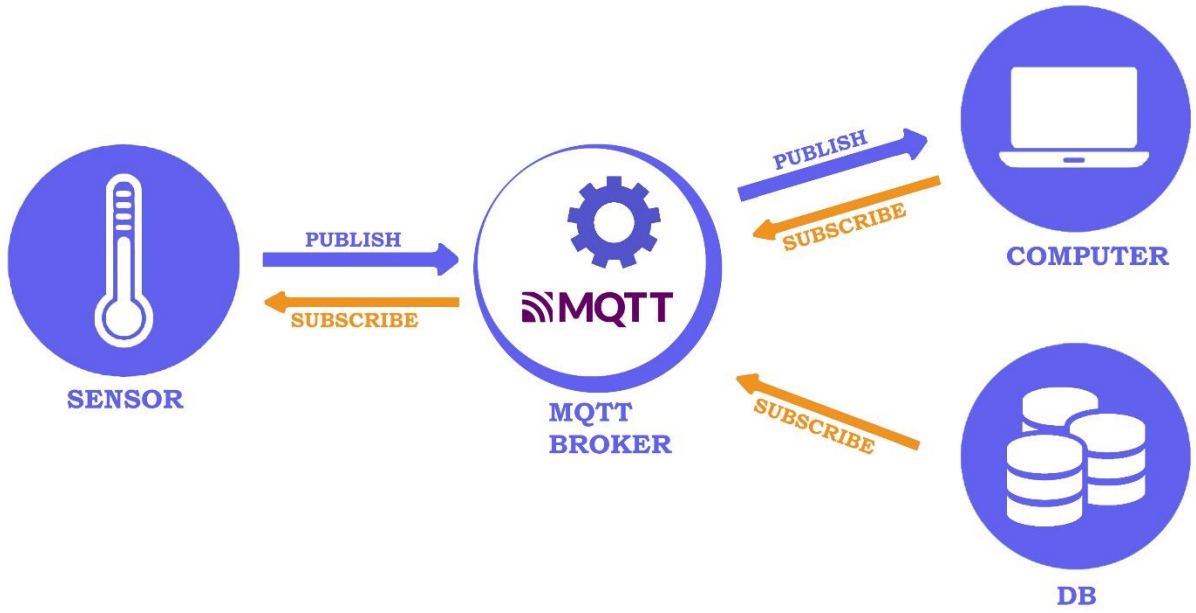
Εικόνα 11: Σύγκριση LoRa – Sigfox- NB-IoT

Τα LPWAN δίκτυα Nb-IoT , Sigfox και LoRa που εξηγήθηκαν παραπάνω χρησιμοποιούνται συχνά σε IoT εφαρμογές , κατά γενική αναλογία προσφέρουν παρόμοιες δυνατότητες και τα χαρακτηριστικά τους εμφανίζουν σημαντικές ομοιότητες. Παρόλα αυτά στη συγκεκριμένη διπλωματική εργασία αποφασίστηκε ο αισθητήριος κόμβος να αποστέλλει τα δεδομένα μέσω LoRa μια και καλύπτει τις ανάγκες σε εμβέλεια, μέγεθος πακέτων και το κόστος υλοποίησης είναι χαμηλό.

Η κάλυψη του δικτύου παρέχεται από την υποδομή του Πανεπιστημίου Δυτικής Μακεδονίας μέσω εγκατεστημένων σταθμών βάσης στην περιοχή του δήμου Κοζάνης.

1.1.3 MQTT

Το MQTT [10] [11] (MQ Telemetry Transport) είναι ένα ελαφρύ πρωτόκολλο δικτύου δημοσίευσης/εγγραφής (publish/subscribe) για μεταφορά μηνυμάτων μεταξύ συσκευών (M2M-Machine to Machine) και εκτελείται μέσω TCP/IP. Είναι ειδικά σχεδιασμένο για συνθήκες τηλεμετρίας περιορισμένου εύρους ζώνης. Αναπτύχθηκε από τους Andy Stanford-Clark (IBM) και Arlen Nipper (Eurotech) το 1999 για τη σύνδεση συστημάτων τηλεμετρίας μέσω δορυφόρων σε αγωγούς πετρελαίου. Παρόλο που ξεκίνησε ως ιδιόκτητο εργαλείο για να επιλύσει κάποια προβλήματα , από το 2010 διατίθεται δωρεάν στο ευρύ κοινό και από το 2014 αποτελεί ανοιχτό πρότυπο OASIS εξυπηρετώντας την ανταλλαγή μηνυμάτων ενός MQTT Client και MQTT Broker.



Εικόνα 12: Πρωτόκολλο επικοινωνίας MQTT

Ο MQTT Client είναι οποιαδήποτε συσκευή (από έναν μικροελεγκτή έως έναν πλήρη διακομιστή) που εκτελεί μια βιβλιοθήκη MQTT και συνδέεται με έναν MQTT Broker μέσω δικτύου. Ο MQTT Broker , συχνά αναφερόμενος ως MQTT Server ,αποτελείται από το λογισμικό που ενεργεί σαν διαμεσολαβητής , εκτελείται είτε σε κάποιον υπολογιστή είτε σε κάποιο Cloud. Ο Broker αναλαμβάνει τον χειρισμό μηνυμάτων μεταξύ της πηγής (Publisher) του μηνύματος και του αποδέκτη (Subscriber). Δεν χρησιμοποιεί την διεύθυνση του παραλήπτη αλλά την γραμμή θέματος «TOPIC», όποιος θέλει ένα αντίγραφο του μηνύματος θα πρέπει να εγγραφεί στο αντίστοιχο TOPIC , πολλοί Clients μπορούν να λάβουν το μήνυμα ενός Broker (ένας προς πολλούς) , και αντίστοιχα πολλοί Publishers μπορούν να δημοσιεύσουν (Publish) θέματα σε έναν Subscriber (πολλοί προς έναν). Το MQTT εφαρμόζει μια αμφίδρομη πολιτική επικοινωνίας η οποία επιτρέπει στον MQTT Client τόσο να παράγει δεδομένα όσο και να λαμβάνει , αυτό πρακτικά σημαίνει πως ένας Client μπορεί να δημοσιεύει (Publish) τις μετρήσεις ενός αισθητήρα σε κάποιο TOPIC ενός Broker και την ίδια στιγμή να μπορεί να λάβει πληροφορίες διαμόρφωσης ή εντολές ελέγχου μέσω ενός TOPIC στο οποίο προηγουμένως έχει εγγραφεί (SUBSCRIBE). Αυτό είναι ιδιαίτερα σημαντικό καθώς βοηθά τόσο στην κοινή χρήση των δεδομένων όσο και στην

διαχείριση και τον έλεγχο των συσκευών. Ένα σημαντικό σημείο του πρωτοκόλλου MQTT παρατηρείται στην αρχιτεκτονική του Broker που καθιστά τις συσκευές πιο ασφαλής εφαρμόζοντας κρυπτογράφηση TLS (Transport Layer Security) με συνδέσεις προστατευμένες από όνομα χρήστη, κωδικό πρόσβασης και προαιρετικά πιστοποιητικά ασφαλείας. Ο εκάστοτε Client δεν γνωρίζει την IP διεύθυνση των υπολοίπων. Ο MQTT Broker έχει τη δυνατότητα να αποθηκεύει τα μηνύματα που λαμβάνει σε κάποια βάση δεδομένων μέσω κάποιου database Client, έτσι ώστε να διατηρούνται τα δεδομένα αλλά και καινούριοι SUBSCRIBERS να έχουν πρόσβαση σε παλαιότερες μετρήσεις. Υπάρχουν αρκετές έτοιμες Cloud λύσεις MQTT Broker που μπορούν να χρησιμοποιηθούν είτε για δοκιμές είτε για πραγματικές εφαρμογές , με πιο διαδεδομένες το Mosquito και το HiveMQ. Τα κύρια πλεονεκτήματα του MQTT Broker είναι πως:

- Εξαλείφει τις ευάλωτες και ανασφαλείς συνδέσεις των CLIENTS.
- Μπορεί εύκολα να εξελιχθεί από μια συσκευή σε ένα ευρύ δίκτυο πολλών συσκευών .
- Έχει τη δυνατότητα να παρακολουθεί και να διαχειρίζεται όλες τις συνδέσεις των CLIENTS παράλληλα με τα διαπιστευτήρια και τα πιστοποιητικά ασφαλείας.
- Επιτυγχάνει μειωμένη πίεση δικτύου χωρίς συμβιβασμούς στην ασφάλεια.

Το MQTT γίνετε ολοένα και πιο διαδεδομένο σε απομακρυσμένες IoT εφαρμογές .Ειδικά σε εφαρμογές έξυπνης γεωργίας όπου είναι απαραίτητο πολλές φορές να θεμελιωθεί ένα δίκτυο από πολλούς κόμβους. Στη συγκεκριμένη διπλωματική εργασία η εγγραφή των δεδομένων του αισθητήριου κόμβου στη βάση δεδομένων γίνεται με τη χρήση MQTT.

1.2 Γλώσσες Προγραμματισμού

Όσον αφορά στις ανάγκες προγραμματισμού του αισθητήριου κόμβου, αναπτύχθηκε κώδικας σε C/C++ καθώς και σε γλώσσα Python για την ανάπτυξη υπηρεσίας αλληλεπίδρασης χρήστη-κόμβου. Τέλος για την προώθηση των δεδομένων μέσω του The Things Network υλοποιήθηκε κατάλληλη συνάρτηση αποκωδικοποίησης, γραμμένη σε JavaScript.

1.2.1 C/C++

Η C είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία αναπτύχθηκε στις αρχές του 1970 από τον Ντένις Ρίτσι στα εργαστήρια BELL της εταιρίας AT&T, είναι μια γλώσσα γενικής χρήσεως και συμβαδίζει με το πρότυπο ANSI. Η C χαρακτηρίζεται από αναγνωσιμότητα (readability) , εύκολη συντήρηση (maintenance) και μεταφερσιμότητα (portability). Η εύκολη ανάγνωση του πηγαίου κώδικα είναι πολύ σημαντική μια και μελλοντικά συμβάλει στην ευκολότερη συντήρηση των προγραμμάτων. Επίσης, προγράμματα γραμμένα σε C είναι εύκολο να μεταφερθούν, είτε μεταξύ διαφορετικών αρχιτεκτονικών είτε μεταξύ διαφορετικών λειτουργικών συστημάτων. Για την σύνταξη ενός προγράμματος σε C ακολουθείται συγκεκριμένο συντακτικό, και τη μεταγλώττιση αναλαμβάνει κατάλληλος μεταγλωττιστής όπως ο GCC.

Η C++ είναι και αυτή μια γλώσσα υψηλού επιπέδου που αναπτύχθηκε στα εργαστήρια της AT&T από τον Δανό Μπιάρνε Στρούστρουπ στις αρχές του 1980. Τα βασικά χαρακτηριστικά και οι κανόνες σύνταξης ενός προγράμματος σε C++ δεν διαφέρουν ιδιαίτερα από τη C μια και αναπτύχθηκε για τη βελτίωση της C. Αποτελεί αντικειμενοστραφή γλώσσα προγραμματισμού που διαθέτει κλάσεις και αντικείμενα και η μεταγλώττιση του πηγαίου κώδικα επιτυγχάνεται με μεταγλωττιστές όπως ο Eclipse και ο GCC.

C	C++
# include <file.h>	# include <file.h>
int x;	Int x;
printf("Hello, world! , enter a number\n");	cout << "Hello World! , enter a number\n";
scanf("%lf , &x);	cin >> x;

for(i = 1 ; i <= 10; i = i+1)
{
printf(" Hello! \n");
}
	for(i = 1 ; i <= 10; i ++)
	{
	cout << "Hello!\n";
	}

if (x < 0) {
printf("enter higher value!\n");
}
	if (x < 0) {
	cout << "enter higher value!\n";
	}

Πίνακας 3: Σύγκριση εντολών C - C++

1.2.2 JavaScript

Η JavaScript (JS) είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού σεναρίων (scripting language) υψηλού επιπέδου που αναπτύχθηκε από τον Μπρένταν Έιτς (Netscape). Είναι μια δυναμική γλώσσα που επιτρέπει ασύγχρονη ανταλλαγή μηνυμάτων , της οποίας οι κανόνες σύνταξης έχουν επηρεαστεί από άλλες γλώσσες υψηλού επιπέδου όπως η C και η C++. Χρησιμοποιείται κυρίως σε web εφαρμογές και παρέχει λύσεις υπολογισμού, επικυρώσεων και χειρισμού δεδομένων. Επί προσθέτως, έχει τη δυνατότητα άμεσης εκτέλεσης μέσω διερμηνευτή , χωρίς να απαιτεί μεταγλώττιση. Η JS είναι μια εξαιρετικά δημοφιλής γλώσσα προγραμματισμού η οποία μαζί με την HTML και τη CSS αποτελούν τις βασικές γλώσσες του προγραμματισμού διαδικτύου. Ακολουθεί παράδειγμα συνάρτησης γραμμένη σε JS, η οποία δέχεται ως είσοδο έναν πίνακα τύπου byte , τον χωρίζει σε δυο μεταβλητής και επιστρέφει αυτές τις μεταβλητές σε μορφή float.

```
function split_byte_message(bytes) {

var message1 = 0 ;
var message2 = 0 ;

for (var i = 0; i < 5; i++) {
  message1 += String.fromCharCode(parseInt(bytes[i]));  }

for (var i = 5; i < bytes.length; i++) {
  message2 += String.fromCharCode(parseInt(bytes[i]));  }

message1 = parseFloat(message1);
message2 = parseFloat(message2);
```

```
return
{ First_message: message1 ,
  Second_Message: message2 ,
};
                                }
```

1.2.3 Python

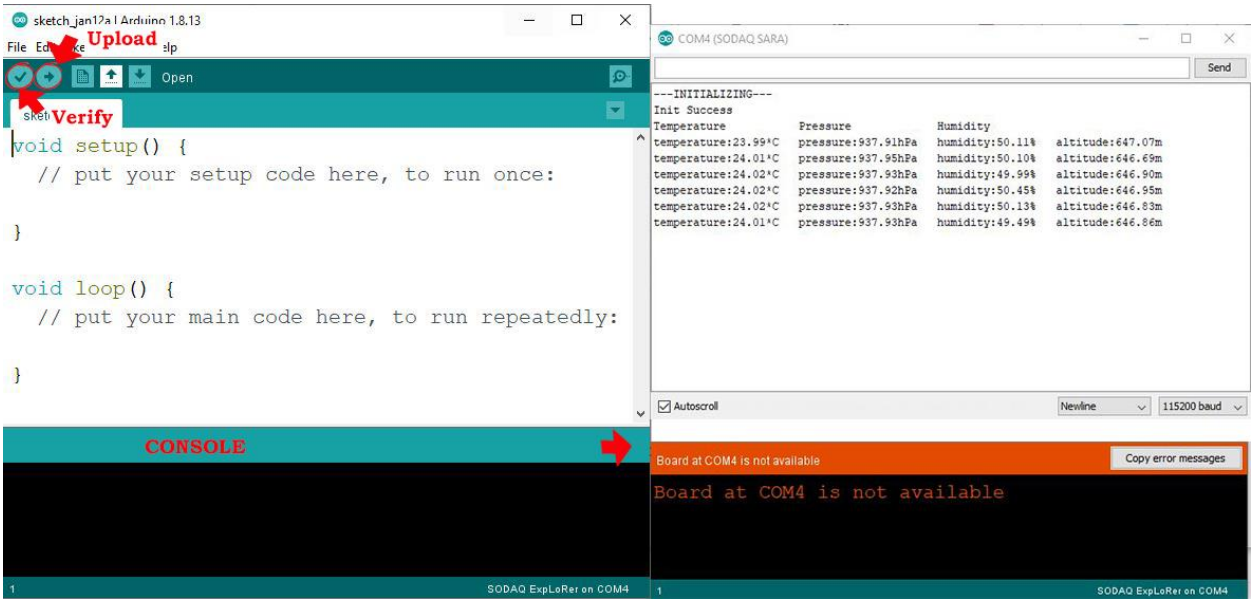
Η Python είναι και αυτή μια γλώσσα προγραμματισμού υψηλού επιπέδου, που αναπτύχθηκε από τον Γκίντο βαν Ρόσσουμ Έιτς (CWI). Υποστηρίζει τον προγραμματισμό σεναρίων (scripting) όπως και η JS και είναι μια δυναμική γλώσσα που εκτελείται με τη χρήση διερμηνευτή. Το βασικό χαρακτηριστικό της Python είναι η προσπάθεια να περιορίσει όσο το δυνατόν τον πηγαίο κώδικα, αντικαθιστώντας για παράδειγμα τη χρήση αγκυλών (brackets – {}) για τον προσδιορισμό ενός block κώδικα με τη χρήση του tab (τεσσάρων κενών). Σε μηχανήματα Linux, η Python συνήθως είναι προ εγκατεστημένη και μπορεί να εκτελεστεί άμεσα με χρήση της εντολής `Python` είτε με την σύνταξη κάποιου script. Ακολουθεί παράδειγμα γραμμένο σε Python που υπολογίζει την περίμετρο ενός τριγώνου και εμφανίζει κατάλληλο μήνυμα στην οθόνη.

```
A = float(input('Give value of the first side'))
B = float(input('Give value of the second side'))
C = float(input('Give value of the third side'))

total = (A + B + C)

print('The perimeter of the triangle is %f' %total)

if total < 10 :
     print('The perimeter is lower than 10')
elif total == 10 :
     print('The perimeter is 10')
else :
     print('The perimeter is greater than 10')
```

1.3 Περιβάλλον προγραμματισμού Arduino IDE

Το περιβάλλον προγραμματισμού Arduino (Arduino Integrated Development Environment – IDE), είναι μια ολοκληρωμένη πλατφόρμα ανοιχτού κώδικα προγραμματισμού μικροελεγκτών Arduino. Επίσης δίνει τη δυνατότητα προγραμματισμού και άλλων μικροελεγκτών μέσω βιβλιοθηκών και οδηγών συσκευών τρίτων κατασκευαστών. Το Arduino IDE διαθέτει επεξεργαστή κειμένου, μέσω του οποίου γίνεται η συγγραφή κώδικα σε γλώσσα wiring η οποία βασίζεται στην C και την C++, επίσης για τη

διαδικασία της μεταγλώττισης χρησιμοποιεί τον GCC. Το λογισμικό διατίθεται δωρεάν σε λειτουργικά Windows, Mac OS και Linux μέσω της επίσημης ιστοσελίδας https://www.arduino.cc/en/software.



Εικόνα 13: Προγραμματιστικό περιβάλλον Arduino IDE

Τα βασικά εργαλεία του Arduino IDE είναι ο επεξεργαστής κειμένου, το serial monitor και η κονσόλα. Μέσω της επιλογής επιβεβαίωσης (verify), ξεκινά η διαδικασία μεταγλώττισης και εφόσον είναι επιτυχής το ανέβασμα του προγράμματος στην εκάστοτε συσκευή γίνεται μέσω του upload. Η κονσόλα είναι πολύ σημαντική καθώς εμφανίζει τυχών σφάλματα κατά τη διαδικασία μεταγλώττισης δίνοντας σημαντικές υποδείξεις για την επίλυση των ζητημάτων. Το serial monitor αποτελεί τον σύνδεσμο μεταξύ του μικροελεγκτή και του υπολογιστή , υποστηρίζει αμφίδρομη επικοινωνία μέσω της οποίας ο προγραμματιστής μπορεί είτε να παρακολουθήσει τη ροή του προγράμματος είτε να ενεργοποιήσει κάποια λειτουργία πληκτρολογώντας μια προκαθορισμένη εντολή. Πριν τον προγραμματισμό μιας συσκευής, είναι σημαντικός ο καθορισμός της συγκεκριμένης πλακέτας και του μικροεπεξεργαστή της καθώς κα η ρύθμιση της συνδεδεμένης σειριακής θύρας. Η προσθήκη βιβλιοθηκών είναι εφικτή είτε από τον Library Manager που διαθέτει πληθώρα βιβλιοθηκών είτε με την προσθήκη κάποιου zip αρχείου.

1.4 Βάσεις δεδομένων χρονοσειρών

Οι βάσεις δεδομένων χρονοσειρών [12] (Time Series Databases – TSDB), είναι ειδικά σχεδιασμένες για το χειρισμό δεδομένων χρονοσειρών. Τέτοια δεδομένα μπορούν να είναι μετρήσεις αισθητήρων , η κατανάλωση ενέργειας και, γενικότερα μεγέθη που χαρακτηρίζονται από τη χρονική στιγμή. Τα δεδομένα σε μια TSDB αποθηκεύονται πάντα σε συνάρτηση με την πάροδο του χρόνου, αυτό έχει ως αποτέλεσμα την ευκολότερη παρακολούθηση της τάσης διακύμανσης είτε αυτή είναι ανοδική είτε καθοδική. Επίσης, ένα ακόμη σημαντικό χαρακτηριστικό των βάσεων δεδομένων χρονοσειρών έχει να κάνει με τη δυνατότητα λήψης συμπερασμάτων σχετικά με την εποχικότητα των μετρήσεων, που σε συνδυασμό με την ανάπτυξη κατάλληλων μοντέλων μπορεί να γίνει εφικτή η δυνατότητα πρόγνωσης ορισμένων συμβάντων. Πιο συγκεκριμένα, στο σενάριο που ένας αισθητήριος κόμβος αποστέλλει μετεωρολογικά δεδομένα για αρκετό χρονικό διάστημα σε μια TSDB και με την εκτέλεση κατάλληλων αλγορίθμων, μια εφαρμογή είναι σε θέση να μπορεί να προβλέψει πως όταν για παράδειγμα μειώνεται η βαρομετρική πίεση την περίοδο του Χειμώνα είναι πολύ πιθανό να επέλθει κακοκαιρία, με βάση τις αποθηκευμένες τιμές της ίδιας εποχής παλαιότερων ετών. Οι βάσεις δεδομένων χρονοσειρών είναι χρήσιμες ακόμη και στην περίπτωση που τα δεδομένα δεν έχουν εποχικό χαρακτήρα αλλά ακολουθούν ανώμαλη διακύμανση, όπως χρηματοοικονομικά δεδομένα.

Οι ακόλουθες βάσεις δεδομένων διαχειρίζονται δεδομένα χρονοσειρών.

Η **Druid** [13] είναι ένα κατανεμημένο σύστημα αποθήκευσης δεδομένων ανοιχτού κώδικα ,γραμμένη σε Java. Η συγκεκριμένη βάση δεδομένων είναι σχεδιασμένη ώστε να απορροφά ταχύτατα μεγάλο όγκο δεδομένων συμβάντων “event data” και να τροφοδοτεί με τα κατάλληλα queries.

Η **eXtremeDB** [14] είναι μια βάση δεδομένων υψηλών αποδόσεων , χαμηλής καθυστέρησης και χρησιμοποιεί αρχιτεκτονική in-memory και ο σχεδιασμός της είναι τέτοιος ώστε να λειτουργεί σε προγράμματα που βασίζονται σε C και C++. Λειτουργεί τόσο σε Windows και Linux όσο και σε άλλα λειτουργικά πραγματικού χρόνου.

Η **kdb+** [15] είναι μια “column-based” σχεσιακή βάση δεδομένων με δυνατότητα αρχιτεκτονικής in-memory όπως και η **eXtremeDB** . Η συγκεκριμένη βάση χρησιμοποιείται συνήθως σε συναλλαγές υψηλής συχνότητας “HFT” , για αποθήκευση , ανάλυση , επεξεργασία αλλά και ανάκτηση μεγάλου όγκου δεδομένων με μεγάλη ταχύτητα. Η **kdb+** μπορεί να χειριστεί δισεκατομμύρια εγγραφές και να αναλύει δεδομένα εντός μια βάσης δεδομένων.

Η **Riak** [16] είναι μια κατανεμημένη βάση δεδομένων που παρέχει υψηλή διαθεσιμότητα, ανοχή σφαλμάτων, λειτουργική απλότητα και επεκτασιμότητα. Η **Riak** εφαρμόζει τις αρχές της Amazon Dynamo με μεγάλες επιρροές του θεωρήματος CAP .Το γεγονός ότι είναι γραμμένη σε Erlang σημαίνει ότι έχει κάποια ανοχή ως προς την αναπαραγωγή και την αυτόματη διανομή των δεδομένων σε όλο το cluster. Διατίθεται τόσο σε έκδοση ανοικτού κώδικα τόσο και σε έκδοση επί πληρωμή.

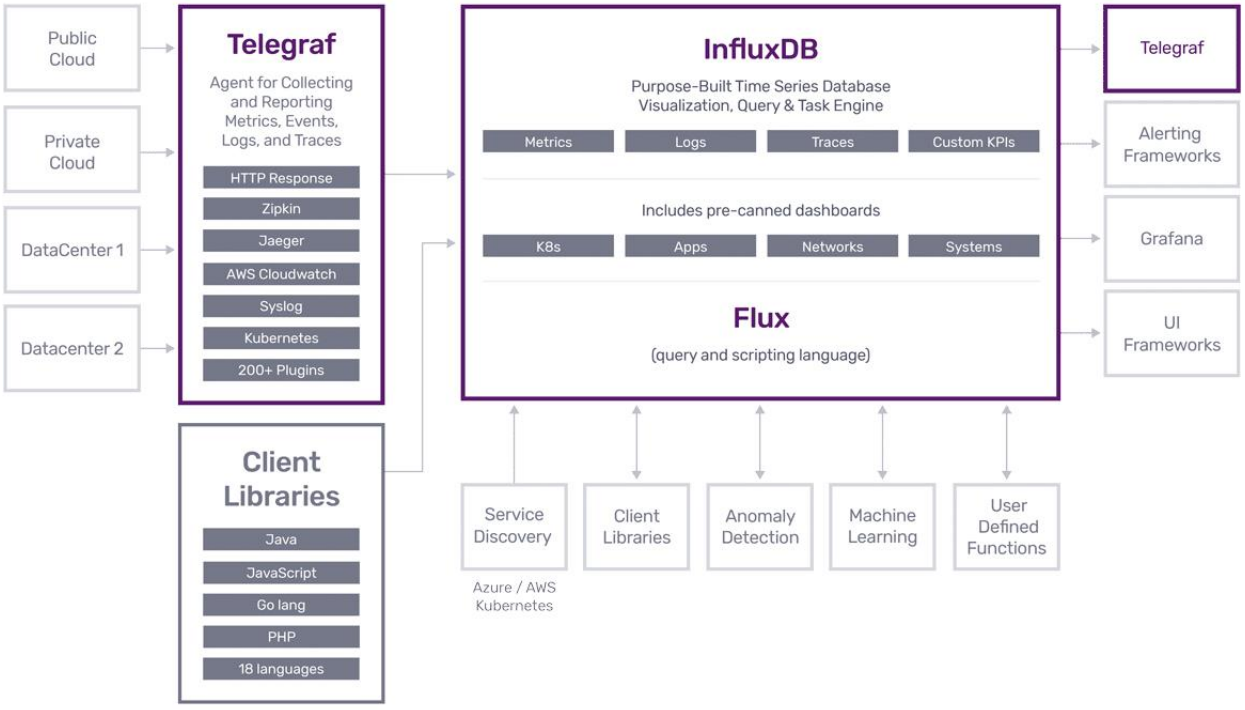
Η **RRDtool** [17] **“**round-robin database too”, είναι μια βάση δεδομένων που στοχεύει στο χειρισμό δεδομένων χρονοσειρών, όπως για παράδειγμα το εύρος ζώνης ενός δικτύου , θερμοκρασίες ή το φορτίο ενός επεξεργαστή. Τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων με κυκλικό buffer έτσι το αποτύπωμα αποθήκευσης του συστήματος παραμένει σταθερό με την πάροδο του χρόνου.

1.4.1 InfluxDB

Για τις ανάγκες αποθήκευσης των δεδομένων μέτρησης του αισθητήριου κόμβου LoRa που υλοποιήθηκε στα πλαίσια της συγκεκριμένης διπλωματικής εργασίας αποφασίστηκε να χρησιμοποιηθεί η βάση δεδομένων χρονοσειρών InfluxDB.

Η InfluxDB [18] είναι μια ανοιχτού κώδικα βάση δεδομένων που αναπτύχθηκε από την InfluxData, είναι γραμμένη σε GO και έχει βελτιστοποιηθεί ώστε να αποθηκεύει και να ανακτά δεδομένα χρονοσειρών. Η χρήση της συναντάτε σε εφαρμογές παρακολούθησης , εφαρμογές μετρήσεων , αλλά και στη συλλογή δεδομένων Iot αισθητήρων πραγματικού χρόνου. Υποστηρίζει πολλαπλά πεδία τιμών, μια αποστολή δεδομένων στη βάση ακολουθεί τη συγκεκριμένη μορφή `measurement-name tag-set field-set timestamp`, για παράδειγμα στην περίπτωση `LoRa_node,host=Telegraf.Autogen,payload_fields_Air_Temperature=23.13 2035218316s` αποθηκεύεται στην βάση δεδομένων, στο πεδίο host=Telegraf.Autogen το μήνυμα payload_fields_Air_Temperature=23.13 τη χρονική στιγμή 2035218316 second προερχόμενο από το LoRa_node. Η InfluxDB μπορεί να συνδυαστεί με διάφορες υπηρεσίες αλλά δεν εξαρτάται από κάποια, εκτελείται στην προκαθορισμένη θύρα 8086 στα πιο κοινά λειτουργικά συστήματα Debian και Windows.

Η InfluxDB πλέον παρέχεται και σαν cloud-based [19] υπηρεσία από την ίδια την InfluxData, η οποία δίνει τη δυνατότητα της μη αναγκαίας χρήση ενός διακομιστή για την εκτέλεσή της καθώς και την πρόσβαση στα δεδομένα από οπουδήποτε. Οι πρόσθετες υπηρεσίες όπως το Telegraf υποστηρίζονται και στην Cloud πλατφόρμα, η τιμολόγηση έχει να κάνει αποκλειστικά με τον όγκο και την χρήση των δεδομένων, ενώ παρέχεται και δωρεάν μέχρι ορισμένο φόρτο εργασίας.



Εικόνα 14: Διάγραμμα δεδομένων InfluxDB

1.5 Εργαλεία και πλατφόρμες

Σε αυτή την ενότητα θα εξηγηθεί η χρήση των πρόσθετων εργαλείων που χρησιμοποιήθηκαν σε αυτή τη διπλωματική εργασία για την προώθηση και την οπτικοποίηση των μετρήσεων στη βάση δεδομένων. Επίσης θα εξηγηθεί η πλατφόρμα ανοιχτού κώδικα « The Things Network » της οποίας η χρήση ήταν πολύ σημαντική.

1.5.1 Chronograf

Το Chronograf [20] αποτελεί το περιβάλλον εργασίας του χρήστη σε μια βάση δεδομένων Influxdb. Είναι ένα εργαλείο εύκολο στη χρήση και εκτελείται στην θύρα 8086. Κύριες λειτουργίες του Chronograf αποτελούν, η οπτικοποίηση, η γενικότερη διαχείριση μιας υποδομής και οι ειδοποιήσεις. Όσον αφορά την οπτικοποίηση, διαθέτει διάφορους προκαθορισμένους πίνακες ελέγχου οι οποίοι με την κατάλληλη τροποποίηση μπορούν να οπτικοποιήσουν άμεσα τα δεδομένα των βάσεων ανάλογα με τον τύπο τους και τις ανάγκες του συστήματος. Το Chronograf σε συνεργασία με το εργαλείο “Capacitor” είναι σε θέση να ελέγχει τα δεδομένα πραγματικού χρόνου και να ενεργοποιεί κατάλληλες ειδοποιήσεις σε περιπτώσεις αποκλίσεων ή μη λήψης δεδομένων, οι ειδοποιήσεις μπορούν να προωθούνται σε διευθύνσεις URL ή απευθείας σε τρίτες εφαρμογές. Τέλος, μέσα από το συγκεκριμένο εργαλείο ο χρήστης είναι σε θέση να εκτελέσει συγκεκριμένα queries για την απομόνωση και τον διαχωρισμό συγκεκριμένων δεδομένων είτε για απευθείας χρήση μέσω των πινάκων ελέγχου είτε για τη χρήση μέσω άλλων εφαρμογών. Το Chronograf δεν αποτελεί απαραίτητο εργαλείο για την εκτέλεση μια Influxdb βάσης δεδομένων ωστόσο είναι ιδανικό για την διαχείριση δεδομένων χρονοσειρών σε αρχική φάση.

1.5.2 Telegraf

Το Telegraf [21] είναι ένα plugin εργαλείο ανοικτού κώδικα για τη λήψη και την προώθηση μετρήσεων σε εφαρμογές στις οποίες εκτελούνται βάσεις δεδομένων και παρέχονται δεδομένα και μετρήσεις , όπως αυτά των IoT αισθητήρων. Το Telegraf είναι γραμμένο σε γλώσσα GO και εκτελείται αυτόνομα με ελάχιστες απαιτήσεις συστήματος. Η ρύθμισή του γίνεται από το αρχείο telegraf.conf, το οποίο δομείται από INPUTS και OUTPUTS. Όσον αφορά τα INPUTS, μπορούν να αποτελούνται από μετρήσεις προερχόμενες απευθείας από συστήματα σε εκτέλεση ή μετρήσεις τρίτων API’s, μπορούν ακόμη να είναι μετρήσεις των υπηρεσιών Kafka και StatsD. Τα OUTPUTS αναλαμβάνουν την αποστολή μετρήσεων σε ένα σύνολο υπηρεσιών όπως βάσεις δεδομένων και MQTT συστήματα. Μπορεί να «γράψει» δεδομένα στην Influxdb προερχόμενα από κάποια πηγή. Στη συγκεκριμένη διπλωματική εργασία ο ρόλος του Telegraf ήταν εξαιρετικά σημαντικός καθώς αποτέλεσε την γέφυρα για τη σύνδεση των μετρήσεων με τη βάση δεδομένων, μέσω της εγγραφής (subscribe) στον MQTT server του The Things Network.

1.5.3 Grafana

Το Grafana [22] αποτελεί λογισμικό ανοικτού κώδικα με σκοπό την οπτικοποίηση και ανάλυση δεδομένων. Επιπλέον, επιτρέπει την εκτέλεση queries καθώς και την δημιουργία ειδοποιήσεων σε δεδομένα χρονοσειρών ανεξάρτητα από την προέλευσή τους. Μετατρέπει τα δεδομένα σε ευανάγνωστα γραφήματα για τον ευκολότερο προσδιορισμό καταστάσεων. Το λογισμικό Grafana με παρόμοιο τρόπο όπως το “Chronograf” δίνει τη δυνατότητα δημιουργίας πινάκων παρακολούθησης με τη χρήση προκαθορισμένων εργαλείων απεικόνισης. Προσφέρει δυνατότητες εξερεύνησης και ανάλυσης

δεδομένων πραγματικού χρόνου αλλά και των αποθηκευμένων στις βάσεις δεδομένων μέσω ad-hoc queries, αυτό δίνει τη δυνατότητα της άμεσης σύγκρισης δεδομένων χρονοσειρών με σκοπό την λήψη σημαντικών συμπερασμάτων σχετικά με την κατάσταση ενός συστήματος με απώτερο σκοπό τη βελτίωση του. Το Grafana διαθέτει υποδομή για αποστολή ειδοποιήσεων σε περιπτώσεις απόκλισης ή υπέρβασης καταχωρημένων ορίων, είναι αξιοσημείωτο το ότι διαθέτει μαθηματικές συναρτήσεις που μπορούν να υπολογίζουν τον ρυθμό αύξησης ή μείωσης ενός μεγέθους. Πρακτικά, σε μια ολοκληρωμένη εφαρμογή έξυπνης γεωργίας λήψης και επεξεργασίας δεδομένων χρονοσειρών όπως μετεωρολογικά δεδομένα, το λογισμικό Grafana είναι σε θέση να αποστέλλει ειδοποίηση μόνο όταν ο ρυθμός αύξησης τείνει να αποκλίνει από τον φυσιολογικό και όχι απλά όταν ένα μέγεθος ξεπεράσει για μια στιγμή τα φυσιολογικά επίπεδα. Αυτό είναι σημαντικό καθώς στο σενάριο αύξησης της ατμοσφαιρικής θερμοκρασίας για μια στιγμή δεν είναι επιθυμητό να αποστέλλεται ειδοποίηση ενώ είναι ιδιαίτερα σημαντικό ο χρήστης να ειδοποιείται στην περίπτωση συνεχόμενης αύξησης ενός τέτοιου μεγέθους.

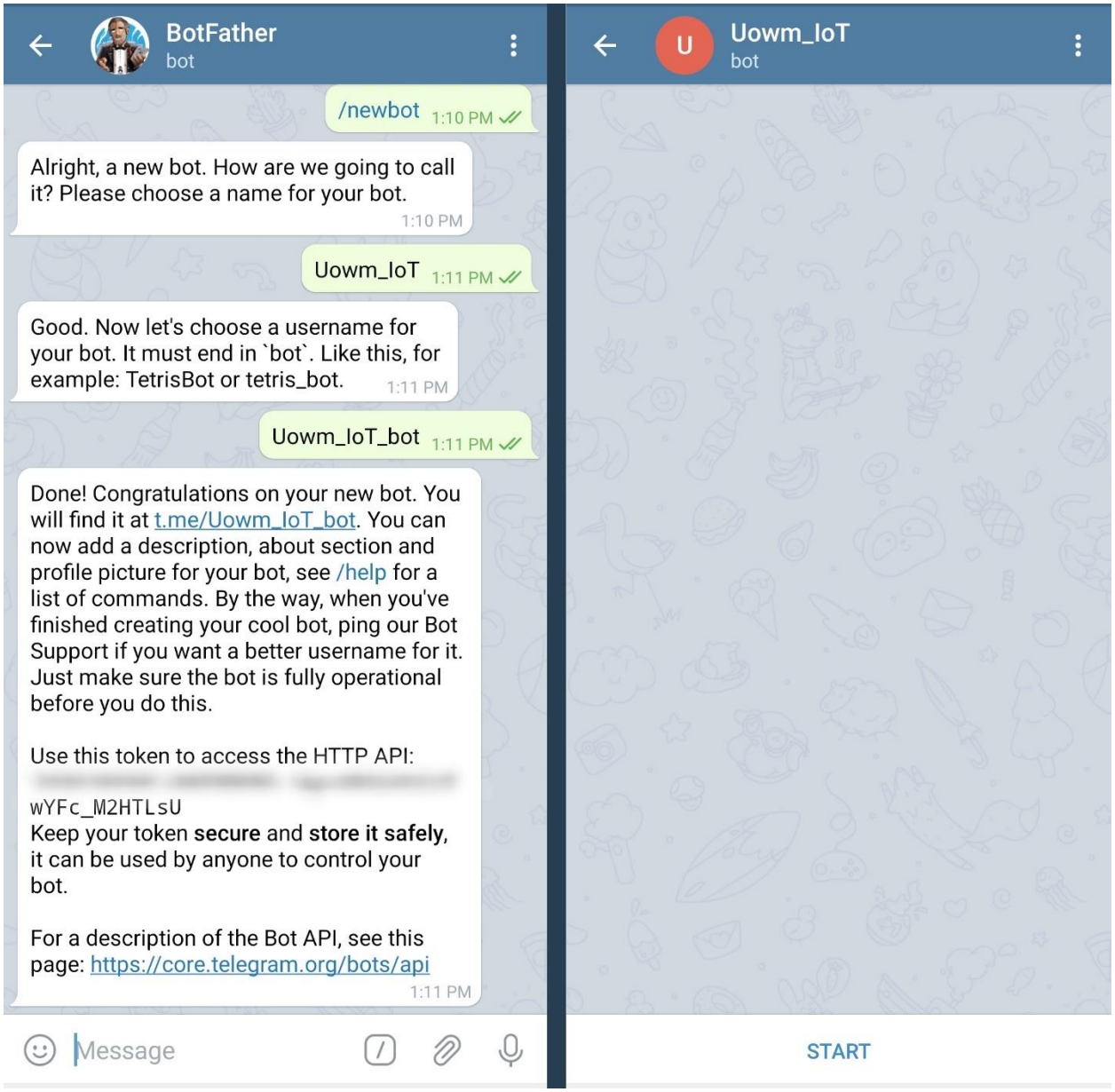
Το Grafana εκτελείται ανεξάρτητα μέσω HTTP στην προκαθορισμένη θύρα 3000 και μπορεί να συλλέγει δεδομένα από διάφορες βάσεις δεδομένων , συμπεριλαμβανομένου της InfluxDB. Τέλος, διατίθεται πληθώρα plugins για την επίλυση ζητημάτων ανάλογα με τις ανάγκες του εκάστοτε συστήματος.

1.5.4 Telegram-bot

Το Telegram [23] αποτελεί ένα δημοφιλές δωρεάν λογισμικό cloud ανταλλαγής μηνυμάτων πραγματικού χρόνου, προσφέρεται τόσο σε κινητές συσκευές Android και iOS όσο και σε συστήματα Windows, macOS και Linux. Σημαντικό στοιχείο του Telegram είναι ότι διαθέτει ισχυρή κρυπτογράφηση για την ανταλλαγή δεδομένων από άκρη σε άκρη, ωστόσο στην συγκεκριμένη διπλωματική εργασία η χρησιμότητα του λογισμικού έγκειται στη δυνατότητα δημιουργίας Telegram bots με σκοπό την άμεση αλληλεπίδραση με τον χρήστη.

Το Telegram από το 2015 δίνει τη δυνατότητα σε τρίτους developers να δημιουργήσουν δωρεάν κάποιο bot και να το ενσωματώσουν στην εφαρμογή τους. Ουσιαστικά πρόκειται για λογαριασμούς Telegram οι οποίο δεν χειρίζονται από κάποιο άτομο αλλά από κατάλληλα υλοποιημένο λογισμικό. Τα bots συχνά διαθέτουν λειτουργίες Ai, ενώ μπορούν να διδάξουν, να παίξουν, να αναζητήσουν, να χειριστούν πληρωμές και να υπενθυμίσουν σχετικά με κάποιο συμβάν. Ο χειρισμός ενός Telegram Bot γίνεται μέσω κάποιου HTTP request ενώ υπάρχουν έτοιμες βιβλιοθήκες τόσο για προγραμματισμό με Python όσο και για προγραμματισμό μέσω κάποιου μικροελεγκτή που έχει την δυνατότητα σύνδεσης στο διαδίκτυο , όπως ο Esp32. Η διαδικασία δημιουργίας ενός Telegram bot περιλαμβάνει με τη σειρά την παρακάτω διαδικασία.

- Αναζήτηση στο Telegram για το BotFather.
- Εντολή /newbot στο κανάλι με το BotFather.
- Δήλωση επιθυμητού ονόματος και ψευδώνυμου στο καινούριο bot.
- Έπειτα το BotFather μας επιστρέφει μήνυμα σχετικά με την επιτυχή δημιουργία του bot στο οποίο περιλαμβάνει το προσωπικό token με το οποίο μπορεί να ενσωματωθεί και να προγραμματιστεί το ίδιο το bot.

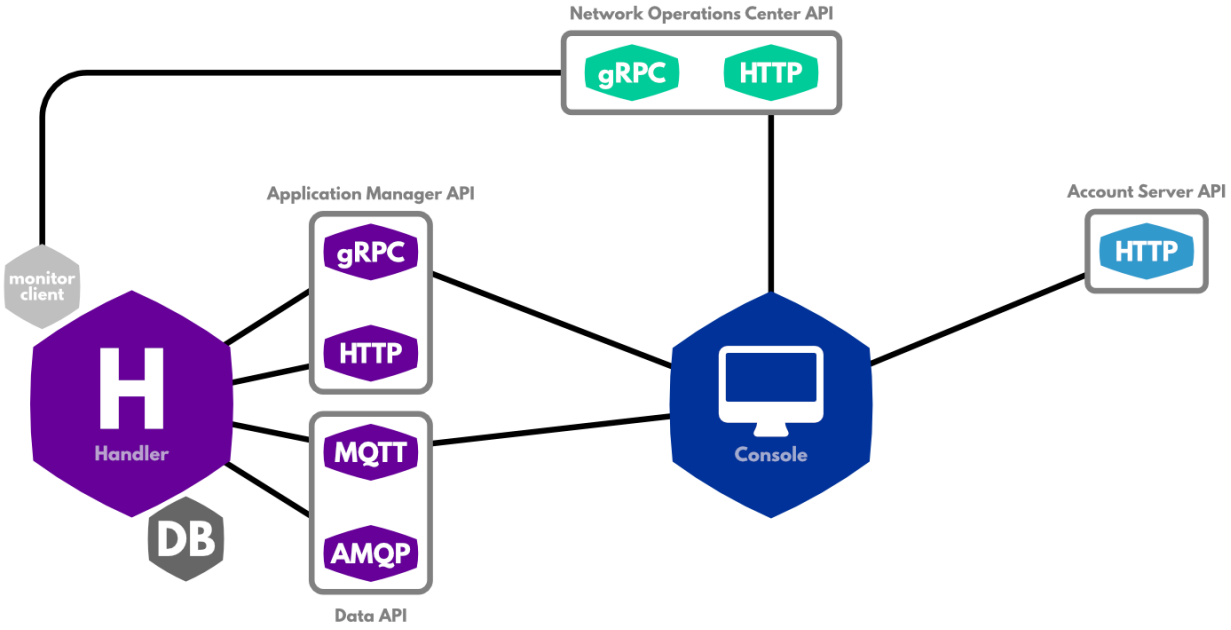


Εικόνα 15: Παράδειγμα Telegram Bot

1.5.5 The Things Network - TTN

Το The Things Network [24] (TTN) αποτελεί μια υποδομή ανοικτού κώδικα η οποία στόχο έχει την ευρύτερη δημιουργία ενός δικτύου βασισμένο σε LoRaWAN με σκοπό την αποστολή και το χειρισμό δεδομένων IoT. Το TTN ξεκίνησε το 2015 στην Ολλανδία από τους Wienke Giezeman και Johan Stokking, από την πρώτη στιγμή η κοινότητα εκτίμησε τις δυνατότητές και τα οφέλη του και μέχρι σήμερα περισσότεροι από εκατό χιλιάδες εθελοντές συμβάλουν στην επέκταση και τη βελτίωση του δικτύου. Οι εθελοντές αναλαμβάνουν την εγκατάσταση και την υποστήριξη σταθμών βάσης LoRaWAN προσπαθώντας να καλύψουν όσο το δυνατόν μεγαλύτερες γεωγραφικές εκτάσεις, από κει και έπειτα εφόσον ένας χρήστης έχει κάλυψη μέσω ενός τουλάχιστον gateway μπορεί να δημιουργήσει μια εφαρμογή για την αποστολή των δεδομένων του στο TTN.

Πρέπει να δοθεί ιδιαίτερη σημασία στο γεγονός ότι η φιλοσοφία της κοινότητας του The Things Network είναι οι υποδομές και η πρόσβαση στο δίκτυο να προσφέρονται δωρεάν στον οποιανδήποτε χρήστη. Μια εφαρμογή στο TTN διαθέτει μοναδικό αναγνωριστικό, το οποίο επιτρέπει τον προσδιορισμό πολλαπλών κόμβων στο δίκτυο. Επίσης μέσω της πλατφόρμας ο χρήστης μπορεί να καταχωρίσει νέες συσκευές σε μια εφαρμογή , καθώς και να πάρει στοιχεία σχετικά με την τρέχουσα κατάστασή τους. Μέσω των μοναδικών διαπιστευτηρίων που παρέχει το TTN, ο χρήστης μπορεί να συνδέσει τις συσκευές του αλλά και να προωθήσει τα δεδομένα του σε τρίτες εφαρμογές. Η παρακολούθηση των δεδομένων που ανταλλάσσει ένας κόμβος με το TTN γίνεται μέσω του application data. Η προώθηση των δεδομένων σε τρίτες εφαρμογές ή βάσεις δεδομένων μπορεί να γίνει ακολουθώντας την φιλοσοφία του MQTT μέσω της υπηρεσίας Node-Red ή στην περίπτωση της συγκεκριμένης διπλωματικής εργασίας μέσω της υπηρεσίας Telegraf.



Εικόνα 16: Κονσόλα TTN

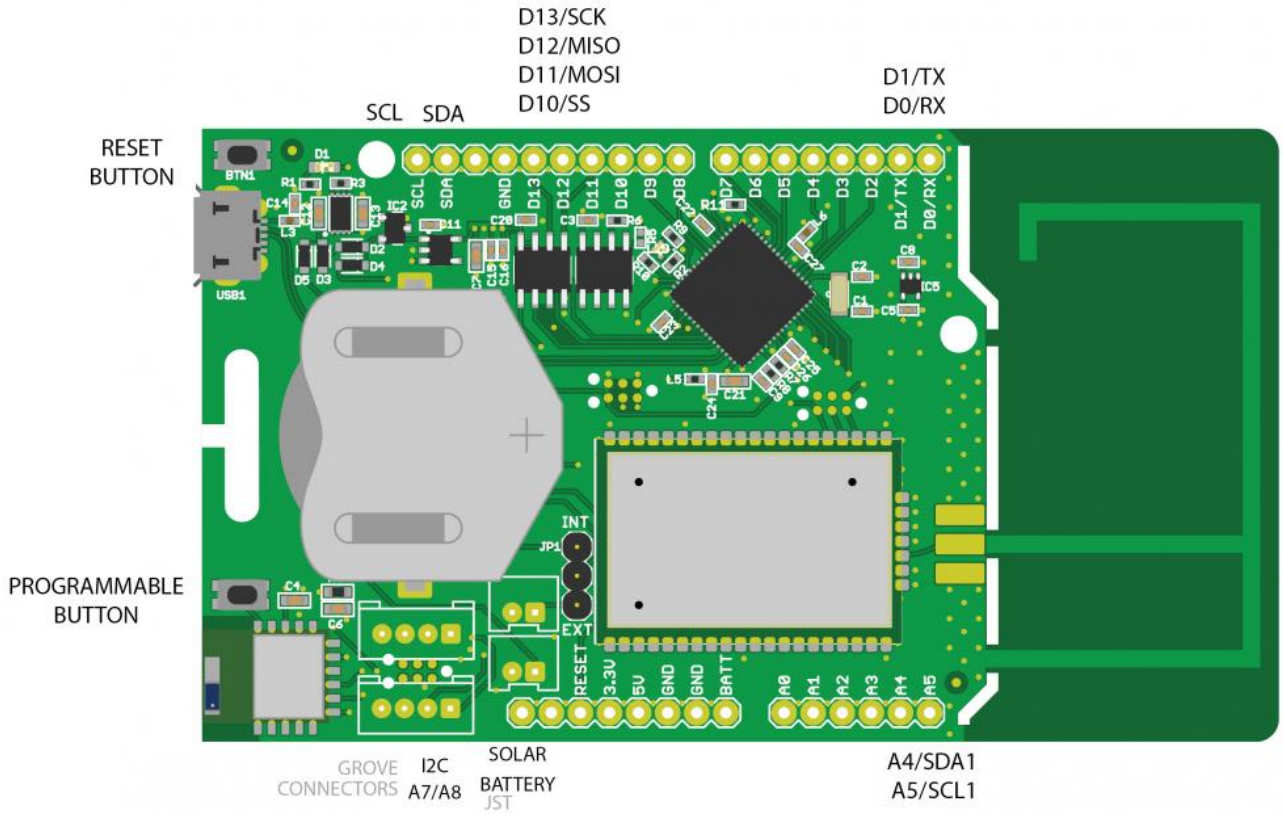
Κεφάλαιο 2ο - Υλοποίηση κόμβου LoRa

2.1 Υλικό μέρος

Το υλικό μέρος που χρησιμοποιήθηκε για την υλοποίηση του αρχικού αισθητήριου κόμβου αποτέλεσαν τα παρακάτω εξαρτήματα:

- SODAQ ExpLoRer
- Waveshare BME 280 Environmental sensor
- Waterproof dallas ds18b20 temperature sensor
- Soil Moisture Sensor YL-69
- Lcd i2c display
- Solar Panel
- Lithium Polymer Li-Po li ion Rechargeable Battery
- CN3065 SOLAR LITHIUM BATTERY CHARGER BOARD
- LM2577 DC-DC 2~24V To 5~28V 2A Volt Step-up Boost Converter

2.1.1 SODAQ ExpLoRer



Εικόνα 17: Διάγραμμα Sodaq ExpLoRer

Το SODAQ ExpLoRer [25] είναι ένα ολοκληρωμένο εργαλείο ανάπτυξης εφαρμογών που προορίζεται κυρίως για εργαστηριακή χρήση έρευνας και ανάπτυξης (R&D) σε εφαρμογές ασυρμάτων δικτύων (LoRa). Τα αξιοσημείωτα χαρακτηριστικά του είναι η ενσωματωμένη κεραία LoRa , που σε

συνδυασμό με το RN2483A της Microchip δίνουν άμεση δυνατότητα στο χρήστη να εξερευνήσει το δίκτυο LoRa, επίσης διαθέτει μπαταρία 3.6v , η οποία σε σχέση με τις ελάχιστες ενεργειακές απαιτήσεις του LoRa μπορεί να τροφοδοτεί τον κόμβο με τάση για μεγάλο χρονικό διάστημα. Δίνεται επίσης η δυνατότητα της απευθείας σύνδεσης με κάποιο ηλιακό πάνελ ή με κάποια μπαταρία 12v καθώς προϋπάρχουν τα απαραίτητα κυκλώματα στην πλακέτα. Τέλος έχει δυνατότητα σύνδεσης μέσω SPI και I2C.

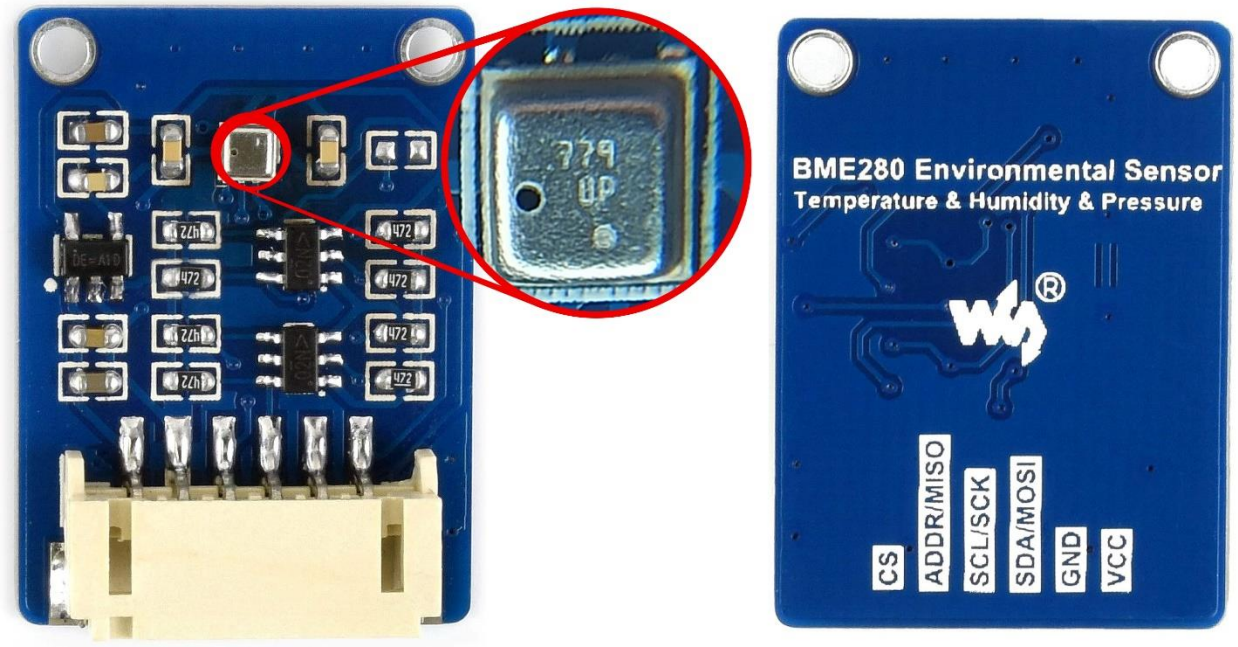
Το SODAQ ExpLoRer βασίζεται στον μικροελεγκτή ATSAMD21J18 32bit, ARM Cortex M0 + , που με την χρήση ενός καλωδίου micro-USB δίνατε εφικτός ο προγραμματισμός του μέσω του Arduino IDE όπως όλους τους κοινούς μικροελεγκτές Arduino.

Τεχνικά Χαρακτηριστικά

Μικροεπεξεργαστής	ATSAMD21J18, 32-bit ARM Cortex M0 +
Συμβατότητα	Συμβατό με Arduino M0
Διαστάσεις	93 x 55 mm
Τάση λειτουργίας	3.3V
Pin Εισόδου/Εξόδου	20
Αναλογικά pin	10-bit DAC
Εξωτερικά INTERUPTS	Σε όλα τα pin
Ρεύμα ανά pin Εισόδου/Εξόδου	7 mA
Μνήμη Flash	256 KB και 4MB (εξωτερική)
SPAM	32KB
Μνήμη EEPROM	Έως 16 KB
Ταχύτητα ρολογιού	48 MHz
Τροφοδοσία	5V USB ή / και 3,7 (της ενσωματωμένης μπαταρίας)
Φόρτιση	έως 500mA
Led	RGB LED, μπλε LED
LoRa	Microchip RN2483 (με ενσωματωμένη κεραία στην pcb ή τοποθέτηση εξωτερικής)
Bluetooth (BLE)	Microchip RN4871
Chip κρυπτογράφησης	ATECC508A
Ενσωματωμένος αισθητήρας θερμοκρασίας	MCP9700AT
Συνδεσιμότητα	MicroUSB
Τύπος ενσωματωμένης μπαταρίας	LIR2450. 3.6V

Πίνακας 4: Χαρακτηριστικά Sodaq ExpLoRer

2.1.2 Waveshare BME 280 Environmental sensor



Εικόνα 18: Αισθητήρας Waveshare BME 280

Το Waveshare BME 280 [26] Environmental sensor , είναι ένα ολοκληρωμένο ψηφιακό module που μετράει ατμοσφαιρική θερμοκρασία , υγρασία και βαρομετρική πίεση. Στην πλακέτα εκτός από τα απαραίτητα ηλεκτρονικά εξαρτήματα συναντάμε έναν μικροσκοπικό ψηφιακό αισθητήρα της εταιρίας BOSCH , ο οποίος είναι υπεύθυνος για τις μετρήσεις. Το συγκεκριμένο module χαρακτηρίζεται από το μικρό του μέγεθος , την υψηλή ακρίβεια καθώς και την σταθερότητά στις μετρήσεις του , για αυτό τον λόγο και ενδείκνυται για IoT εφαρμογές. Υποστηρίζει διασύνδεση I2C και SPI το οποίο έχει ως αποτέλεσμα να μπορεί να συνδυαστεί με Raspberry Pi/Arduino/STM32.

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	5V/3.3V
Διασύνδεση	I2C/SPI
Εύρος θερμοκρασίας	-40-85°C (Ακρίβεια 0.1°C, ανοχή ±1°C)
Εύρος Υγρασίας	0-100%rh (Ακρίβεια 0.008%RH, ανοχή ±1°C)
Εύρος Πίεσης	300-1100hPa(Ακρίβεια 0.18Pa, ανοχή ±1hPa)
Διαστάσεις	27mm x 20mm

Πίνακας 5: Χαρακτηριστικά BME 280

2.1.3 Waterproof dallas ds18b20 temperature sensor

Εικόνα 19: Αισθητήρας dallas ds18b20

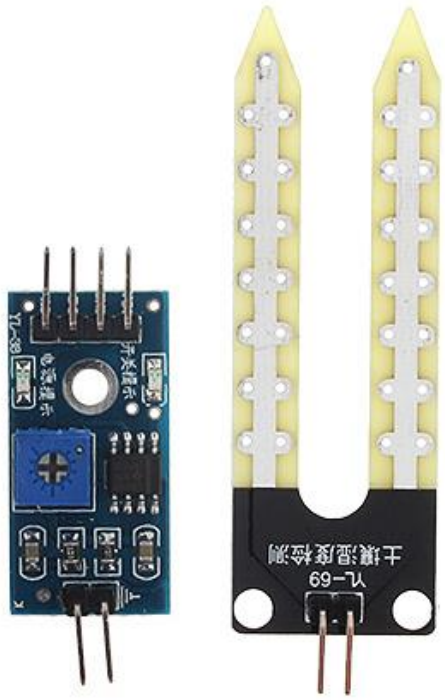
Το waterproof dallas ds18b20 [27] είναι ένα αδιάβροχο ψηφιακό αισθητήριο , κατάλληλο για χρήση σε χώμα ή νερό. Διαθέτει 3 καλώδια , 2 για την τροφοδοσία (VDD, GND) και ένα για την αποστολή δεδομένων (DATA). Ο αισθητήρας έχει τη δυνατότητα να αποστέλλει ένα ψηφιακό σήμα σε κάποιον μικροελεγκτή σε οποιοδήποτε ψηφιακό pin , ακρίβειας 9 και 12bit σε 93,75 και 750ms αντίστοιχα.
Εξαιτίας του χαρακτηρισμού του ds18b20 από μοναδικό ID , πολλοί αισθητήρες του ίδιου τύπου μπορούν να στέλνουν δεδομένα στην ίδια ψηφιακή είσοδο. Επίσης ο συγκεκριμένος αισθητήρας λόγω της ψηφιακής του διασύνδεσης , έχει τη δυνατότητα να συνδέεται αρκετά μακριά από τον μικροελεγκτή χωρίς να μειώνεται η ποιότητα της μέτρησης.

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	3-5V
Διασύνδεση	Ενός ψηφιακού καλωδίου
Εύρος θερμοκρασίας	55°C έως +125°C
Ακρίβεια μέτρησης	±0.5°C (στο εύρος -10°C έως +85°C)
Ακρίβεια δεδομένων	9bit -12bit
Μνήμη	64-bit lithography ROM
Χρόνος απόκρισης	Λιγότερο από 750ms

Πίνακας 6: Χαρακτηριστικά dallas ds18b20

2.1.4 Soil Moisture Sensor YL-69



Εικόνα 20: Αισθητήρας YL-69

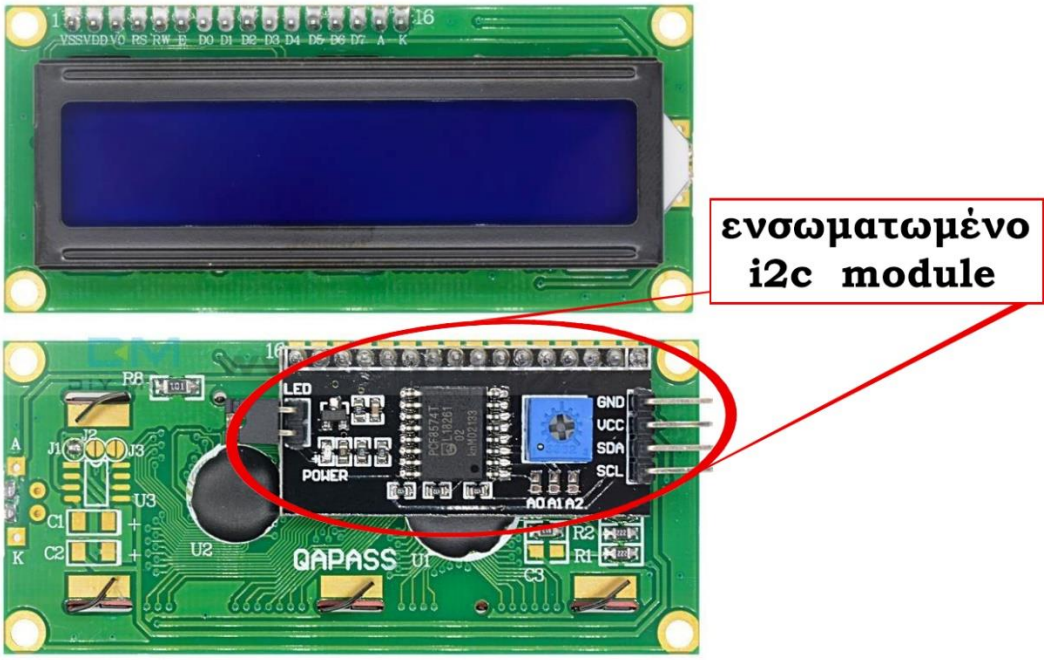
Το YL-69 [28] είναι ένα αισθητήριο που χρησιμοποιείται για την ανίχνευση της υγρασίας του εδάφους. Είναι ιδανικό για IoT εφαρμογές παρακολούθησης καλλιεργειών και αυτοματισμών. Ο αισθητήρας αποτελείται από δυο μέρη , από τον ανιχνευτή-probe (Yl-69) , και την ηλεκτρονική πλακέτα (Yl-39) η οποία συνδέεται με τον εκάστοτε μικροελεγκτή. Έχει τη δυνατότητα να αποστέλλει είτε αναλογικά είτε ψηφιακά σήματα. Πρακτικά ο αισθητήρας ανιχνεύει την αντίσταση που αντιμετωπίζει το ηλεκτρικό ρεύμα καθώς ρέει από το ένα άκρο του ανιχνευτεί προς το άλλο . Περισσότερη υγρασία καθιστά το έδαφος πιο αγώγιμο. Όταν τα επίπεδα υγρασίας είναι χαμηλά η τάση είναι υψηλή ενώ όταν παρατηρείται υψηλή υγρασία μειώνεται η τάση εξόδου αντίστοιχα. Τέλος διαθέτει ενσωματωμένο ποτενσιόμετρο για τη ρύθμιση της ευαισθησίας.

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	3.3-5V
Διασύνδεση	Ενός ψηφιακού pin (D0) / Ενός αναλογικού pin (AO)
Αναλογικό σήμα εξόδου	0 – 4.2V
Comparator chip	LM393
Εύρος υγρασίας	0-100%rh
Ακρίβεια	Εξαρτάτε από τις συνθήκες , κυμαίνεται μεταξύ ~1-10%

Πίνακας 7: Χαρακτηριστικά YL-69

2.1.5 Lcd i2c display



Εικόνα 21: Οθόνη i2c υγρών κρυστλαλλων 2x16

Η συγκεκριμένη οθόνη , είναι μια οθόνη υγρών κρυστάλλων (Liquid Crystal Display – LCD) η οποία είναι συμβατή με Arduino συσκευές μέσω i2c με τη χρήση κατάλληλης βιβλιοθήκης. Την επικοινωνία μέσω του πρωτοκόλλου i2c αναλαμβάνει το ενσωματωμένο module το οποίο δίνει στην lcd την διεύθυνση 0x27. Υποστηρίζει 16 χαρακτήρες ανά γραμμή στις συνολικά 2 γραμμές που διαθέτει , οι χαρακτήρες εμφανίζονται με άσπρο χρώμα και ο οπίσθιος φωτισμός της συσκευής (backlight) είναι μπλε χρώματος, τέλος διαθέτει ποτενσιόμετρο για την ρύθμιση της αντίθεσης.

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	5V
Διασύνδεση	I2c
Σύνολο χαρακτήρων	16 x 2
Προκαθορισμένη διεύθυνση	0x27
Φωτισμός	Χαρακτήρες-άσπρο χρώμα , backlight-μπλε
Διαστάσεις	80 x 36 x 20mm

Πίνακας 8: Χαρακτηριστικά οθόνης LCD

2.1.6 Solar Panel

Εικόνα 22: Φωτοβολταϊκή κυψέλη 6V

Το συγκεκριμένο OEM φωτοβολταϊκό πάνελ χρησιμοποιήθηκε για της ανάγκες τροφοδοσίας του αισθητήριου κόμβου, χάρη στο οποίο δίνεται η δυνατότητα στο SODAQ ExpLoRer να παρέχει εκτός από 3.3V και 5V , στις συνδεδεμένες συσκευές. Επίσης αναλαμβάνει την φόρτιση της ενσωματωμένης καθώς και της εξωτερικής μπαταρίας. Όσον αφορά το συγκεκριμένο πάνελ , στην πραγματικότητα πρόκειται για ένα στοιχείο των 6v από πολυκρυσταλλικό πυρίτιο με πολύ μικρό μέγεθος που μπορεί να αποδώσει έως και 2Watt.

Τεχνικά Χαρακτηριστικά

Τάση Εξόδου	6V (~7V σε κατάσταση ανοικτού κυκλώματος)
Ισχύς	2W/p
Ένταση Ρεύματος εξόδου	~350mA
Στεγανότητα	IP65
Διαστάσεις	110 x 60 x 2.5 mm
Κυψέλες	12 κυψέλες (0.5V / Κυψέλη)

Πίνακας 9: Χαρακτηριστικά Φ/Β πάνελ

2.1.7 Lithium Polymer Li-Po li ion Rechargeable Battery

Εικόνα 23: Επαναφορτιζόμενη μπαταρία λιθίου 3.7V

Για της ανάγκες τροφοδοσίας ηλεκτρικού ρεύματος του αισθητήριου κόμβου χρησιμοποιήθηκε και η συγκεκριμένη επαναφορτιζόμενη μπαταρία λιθίου li-ion . Η τάση εξόδου της είναι 3.7V και η συνολική της χωρητικότητα είναι 1200mAH.

Τεχνικά Χαρακτηριστικά

Μοντέλο	HS1364
Τάση εξόδου	3.7V
Χωρητικότητα	1200mAH
Μέγιστη τάση φόρτισης	~4.2V
Κύκλοι φόρτισης	>500 / Έτος
Διαστάσεις	35 x 25 x 6 mm

Πίνακας 10: Χαρακτηριστικά μπαταρίας λιθίου

2.1.8 CN3065 SOLAR LITHIUM BATTERY CHARGER BOARD

Εικόνα 24: Φορτιστής ηλιακών πάνελ CN3065

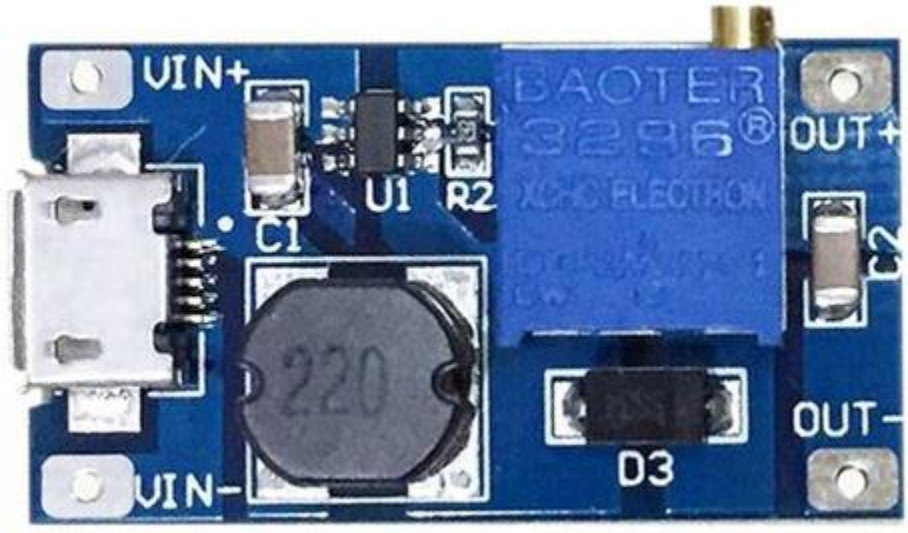
Το CN3065 SOLAR LITHIUM BATTERY CHARGER BOARD αποτελεί έναν φορτιστή ηλιακών πάνελ εξαιρετικά μικρού μεγέθους. Αναλαμβάνει την φόρτιση μια μπαταρίας Li-Po με μέγιστο ρεύμα φόρτισης τα 500ma και είσοδο από το φ/β πάνελ τα 6V. Διαθέτει ένδειξη φόρτισης καθώς κύκλωμα προστασίας βραχυκυκλώματος, ενώ σταθεροποιεί την τάση εξόδου.

Τεχνικά Χαρακτηριστικά

Τσιπ	CN3065
Τάση εισόδου φ/β πάνελ	4.4-6V
Μέγιστο ρεύμα φόρτισης	500mA
Συνδεσιμότητα	JST 2mm και microUSB
Ένδειξη LED	Κόκκινο-φόρτιση, Πράσινο-φορτισμένο
Συμβατές μπαταρίες	Ιόντων Λιθίου Li-Po

Πίνακας 11: : Χαρακτηριστικά CN3065

2.1.9 LM2577 DC-DC 2~24V To 5~28V 2A Volt Step-up Boost Converter



Εικόνα 25: Μετατροπέας step up 2~24 --> 5~28 Volt

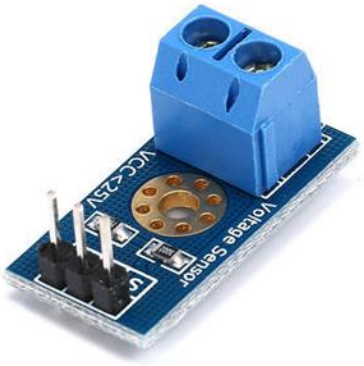
Το LM2577 DC-DC 2~24V To 5~28V 2A Volt Step-up Boost Converter είναι ένα ολοκληρωμένο εργαλείο μετατροπής-ενίσχυσης της τάσης εισόδου (step up). Δέχεται ως είσοδο μια τάση συνεχόμενου ρεύματος από 2 έως 24V και μπορεί να το αυξήσει στα 5 έως 28V με μέγιστο ρεύμα εξόδου τα 2A. Η ενίσχυση επιτυγχάνεται μέσω ειδικά σχεδιασμένου κυκλώματος συνδυάζοντας το IC LM2577 με κατάλληλους πυκνωτές, διόδους και αντιστάσεις.

Τεχνικά Χαρακτηριστικά

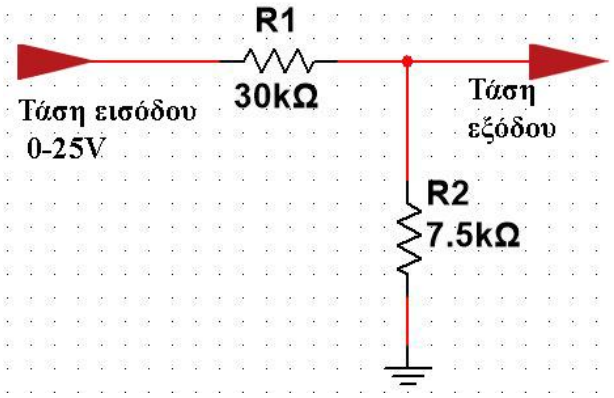
Τσιπ	LM2577
Τάση εισόδου	2-24V
Τάση εξόδου	5-28V
Μέγιστο ρεύμα εξόδου	2A
Απόδοση	Έως 93%
Συνδεσιμότητα	Επαφές κόλλησης και USB

Πίνακας 12: : Χαρακτηριστικά LM2577

2.1.10 Voltage detection module 0-25V



Εικόνα 26: Voltage detection module



Εικόνα 27: Ισοδύναμο κύκλωμα voltage detection module

Το συγκεκριμένο module είναι πρακτικά ένας απλός διαιρέτης τάσης 5:1, όπως φαίνεται και στο παραπάνω κύκλωμα αποτελείται από δύο αντιστάσεις, μια των 30k ohm και μια των 7.5k ohm. Υποβαθμίζει μια τάση συνεχόμενου ρεύματος από το εύρος 0 – 25V στο εύρος 0 – 5V, την οποία μπορούν να διαβάζουν οι μικροελεγκτές (όπως το Sodaq ExpLoRer) μέσω ενός αναλογικού pin. Το συγκεκριμένο module χρησιμοποιείται για την εποπτεία της τάσης του κυκλώματος ηλιακής φόρτισης.

Τεχνικά Χαρακτηριστικά

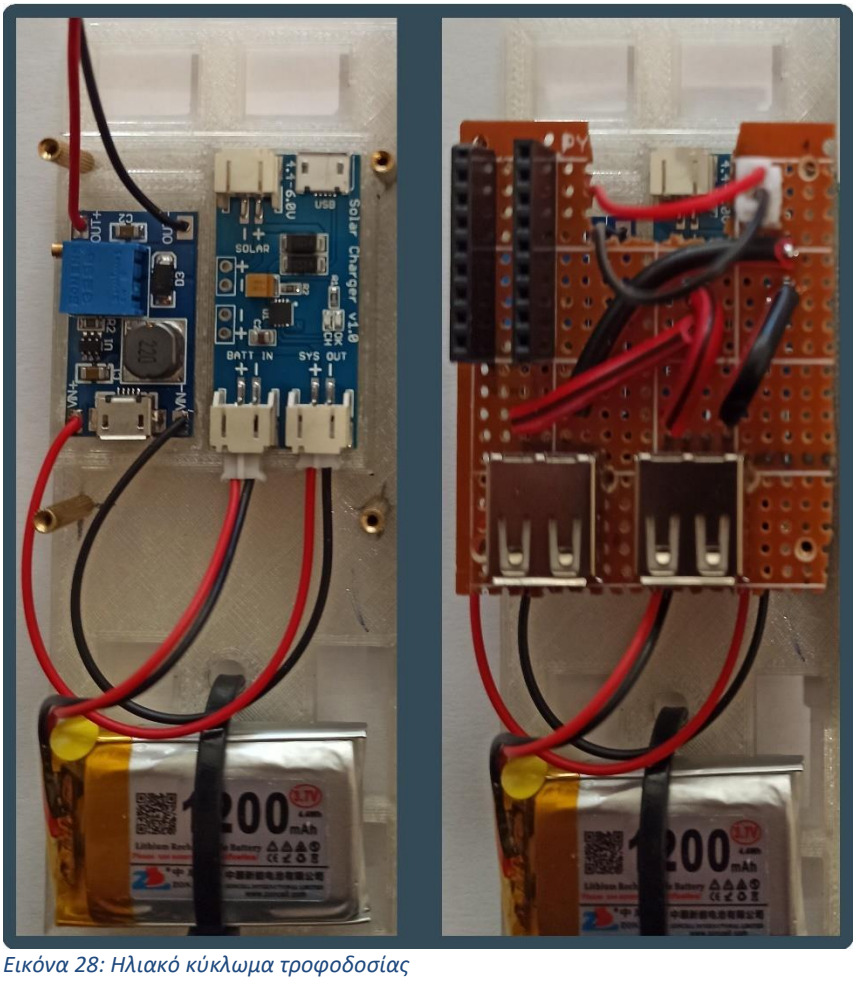
Τάση εισόδου	0-25V
Τάση εξόδου	0-5V
Ακρίβεια	0.00489 V
Συνδεσμολογία Εισόδου	**+** → 5/3.3V , **-** → GND, **S** → Analog input
Συνδεσμολογία Εισόδου	**+** → VCC, **-** → GND

Πίνακας 13: : Χαρακτηριστικά voltage detection module

2.2 Τροφοδοσία – Κύκλωμα ηλιακής φόρτισης

Όσον αφορά την τροφοδοσία του αισθητήριου κόμβου, αρχικά έγιναν δοκιμές ώστε το φωτοβολταϊκό πάνελ και η εξωτερική μπαταρία να συνδεθούν απευθείας στο ενσωματωμένο κύκλωμα φόρτισης του SODAQ ExpLoRer, αυτό είχε ως αποτέλεσμα να μην εφικτή η τροφοδοσία συσκευών 5V παρά μόνο 3.3V. Για αυτό το λόγο αποφασίστηκε να υλοποιηθεί ξεχωριστό κύκλωμα φόρτισης και τροφοδοσίας του SODAQ ExpLoRer ώστε ο αισθητήριος κόμβος να είναι εύκολα επεκτάσιμος και να μπορεί να υποστηρίξει και συσκευές 5V.

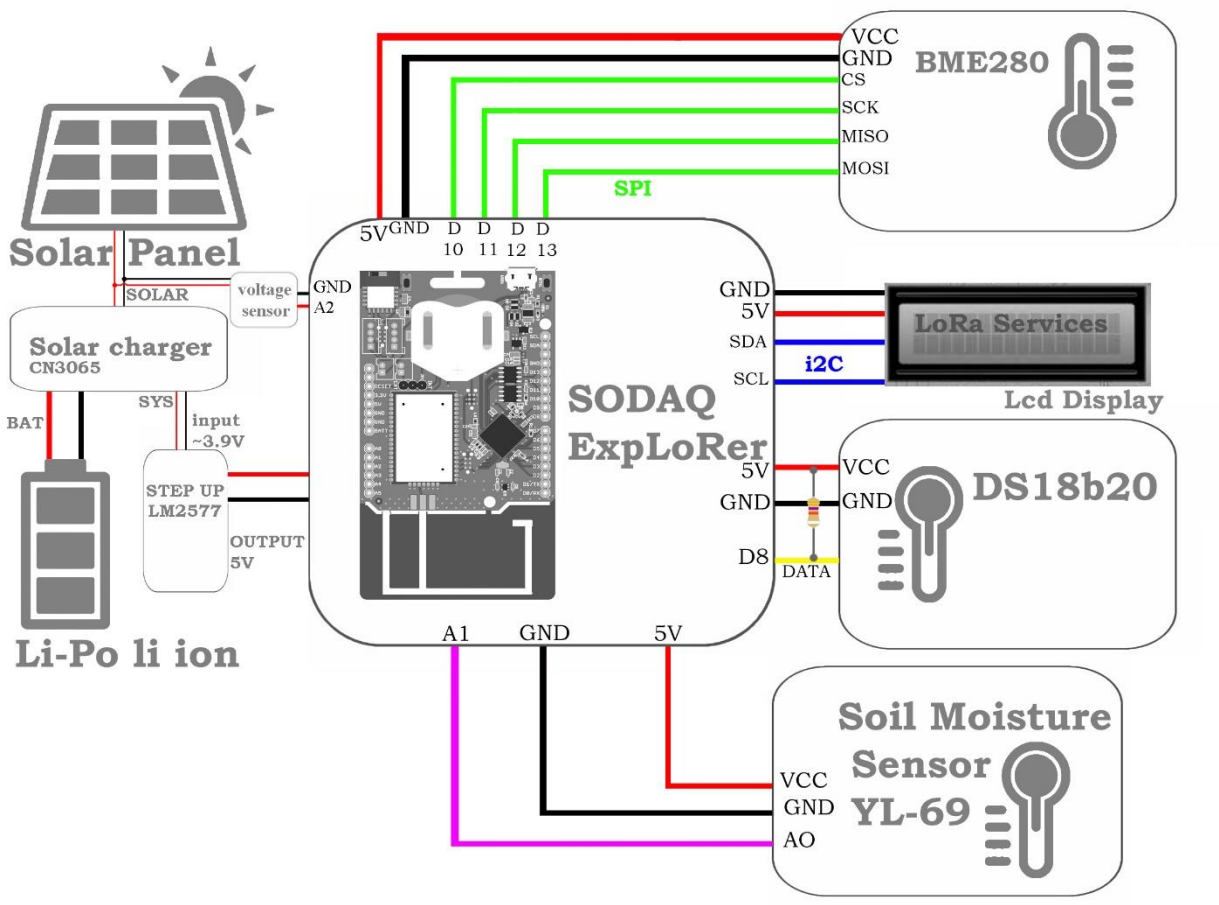
Για την υλοποίηση του ξεχωριστού κυκλώματος φόρτισης χρησιμοποιήθηκε η πλακέτα φόρτισης CN3065 SOLAR LITHIUM BATTERY CHARGER καθώς και ο μετατροπέας LM2577 Step-up Boost Converter. Σχετικά με τη συνδεσμολογία του κυκλώματος, ο φορτιστής συνδέεται με το φ/β πάνελ στη θέση SOLAR και με την εξωτερική μπαταρία λιθίου στη θέση BAT. Με βάση την παραπάνω σύνδεση ο φορτιστής είναι σε θέση να αναγνωρίζει την ανάγκη φόρτισης της μπαταρίας και να κρατά σταθερή την τάση εξόδου στα 3.9V στη θέση SYS. Έπειτα ακολουθεί η πλακέτα LM2577 Step-up Boost Converter, η οποία συνδέεται στη θέση SYS με την τάση εισόδου της πλακέτας φόρτισης και μετά από κατάλληλη ρύθμιση του ενσωματωμένου ποτενσιόμετρου επιστρέφει σταθερή τάση 5V. Τέλος παρεμβάλεται μια custom pcb πλακέτα στην οποία τα 5V διατίθενται με τη μορφή των κλασικών 1-pin socket αλλά και me USB.



Εικόνα 28: Ηλιακό κύκλωμα τροφοδοσίας

2.3 Συνδεσμολογία

Η συνδεσμολογία που ακολουθήθηκε για την υλοποίηση του αισθητήριου κόμβου LoRa φαίνεται αναλυτικά στο παρακάτω σχηματικό διάγραμμα. Αξίζει να αναφερθεί πως όσον αφορά την σύνδεση του αισθητήρα DS28b20 για την λήψη της θερμοκρασίας εδάφους , παρεμβάλεται μια αντίσταση 4.7ΚΩ ανάμεσα στην τροφοδοσία 5v και στον αγωγό των δεδομένων (data). Στην πραγματικότητα πρόκειται για μια αντίσταση pull up (pull-up resistor) , της οποίας η χρήση είναι απαραίτητη για να παρέχει μια τιμή αναφοράς στον μικροελεγκτή όταν το κύκλωμα είναι ανοικτό , η τιμή των 4.7 Ομ είναι ιδανική καθώς επιτρέπει μικρό ρεύμα να φτάνει στην είσοδο D8 .



Εικόνα 29: Συνδεσμολογία αισθητήριου κόμβου

2.4 Αισθητήριος κόμβος

Ο τελικός προορισμός του αισθητήριου κόμβου LoRa προορίζεται για εξωτερική χρήση σε εκτάσεις γης επομένως θα πρέπει να είναι ανθεκτικός στις καιρικές συνθήκες και όσο το δυνατόν πιο φορητός χωρίς συνέπειες στην συνολική του ποιότητα, δηλαδή δεν θα πρέπει να μεταβάλλεται η ποιότητα των μετρήσεων και η αποστολή των δεδομένων. Για αυτούς τους λόγους η πλακέτα SODAQ ExpLoRer μαζί με την μπαταρία τοποθετήθηκαν σε στεγανό ηλεκτρολογικό κουτί από πλαστικό (IP55), ενώ με τη χρήση τρισδιάστατου εκτυπωτή εκτυπώθηκαν απαραίτητα εξαρτήματα για την στεγανοποίηση των αισθητήρων και την τοποθέτηση του κόμβου.

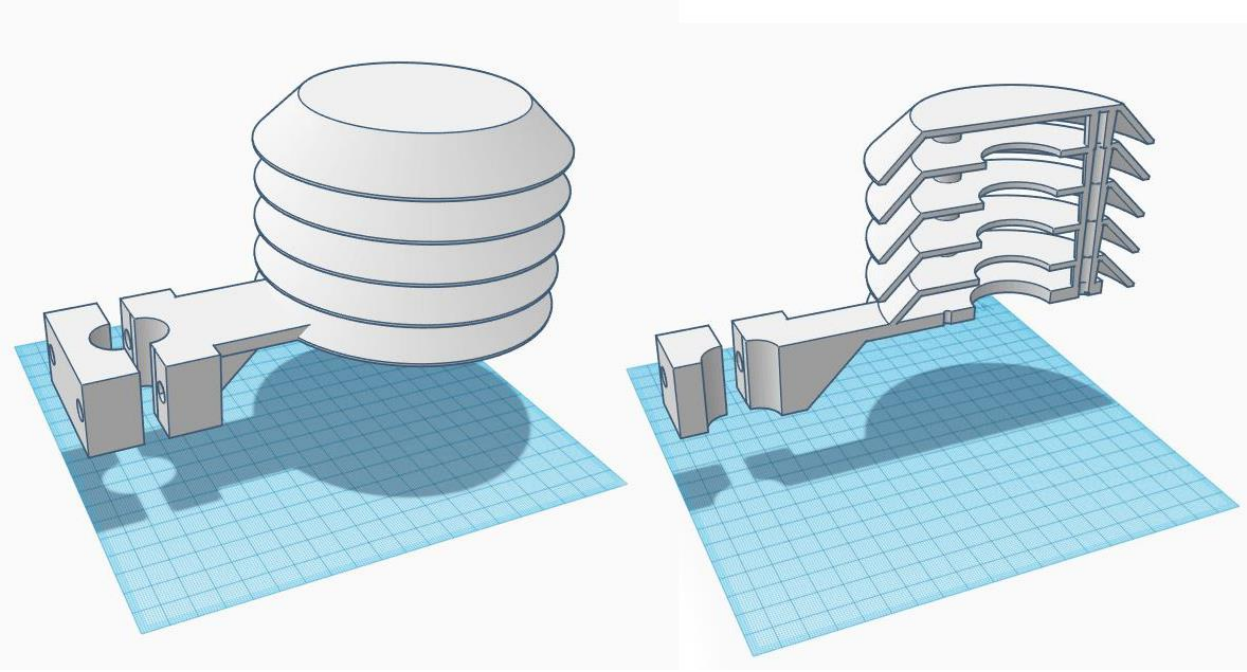
Εικόνα 30: Τοποθέτηση Sodaq ExpLoRer και κυκλώματος ηλιακής φόρτισης

2.4.1 Σχέδια τρισδιάστατων εκτυπώσεων

Οι τρισδιάστατες εκτυπώσεις έγιναν με τη χρήση ενός Creality 3D Ender-3 V2, αρχικά το υλικό εκτύπωσης που χρησιμοποιήθηκε ήταν νήμα PLA (πολυγαλακτικό οξύ). Το PLA είναι βιοδιασπώμενο υλικό φυτικής προέλευσης , δεν ενδείκνυται για εξωτερική χρήση αλλά αποτελεί την καλύτερη λύση για την κατασκευή πρωτότυπων μοντέλων καθώς είναι εύκολο στην επεξεργασία λόγω της δυνατότητάς του να εκτυπώνεται σε μικρές θερμοκρασίες. Για κατασκευή μοντέλων που εκτίθενται στις καιρικές συνθήκες προτιμάται η χρήση υλικών όπως το PETG, το ABS και το ASA.

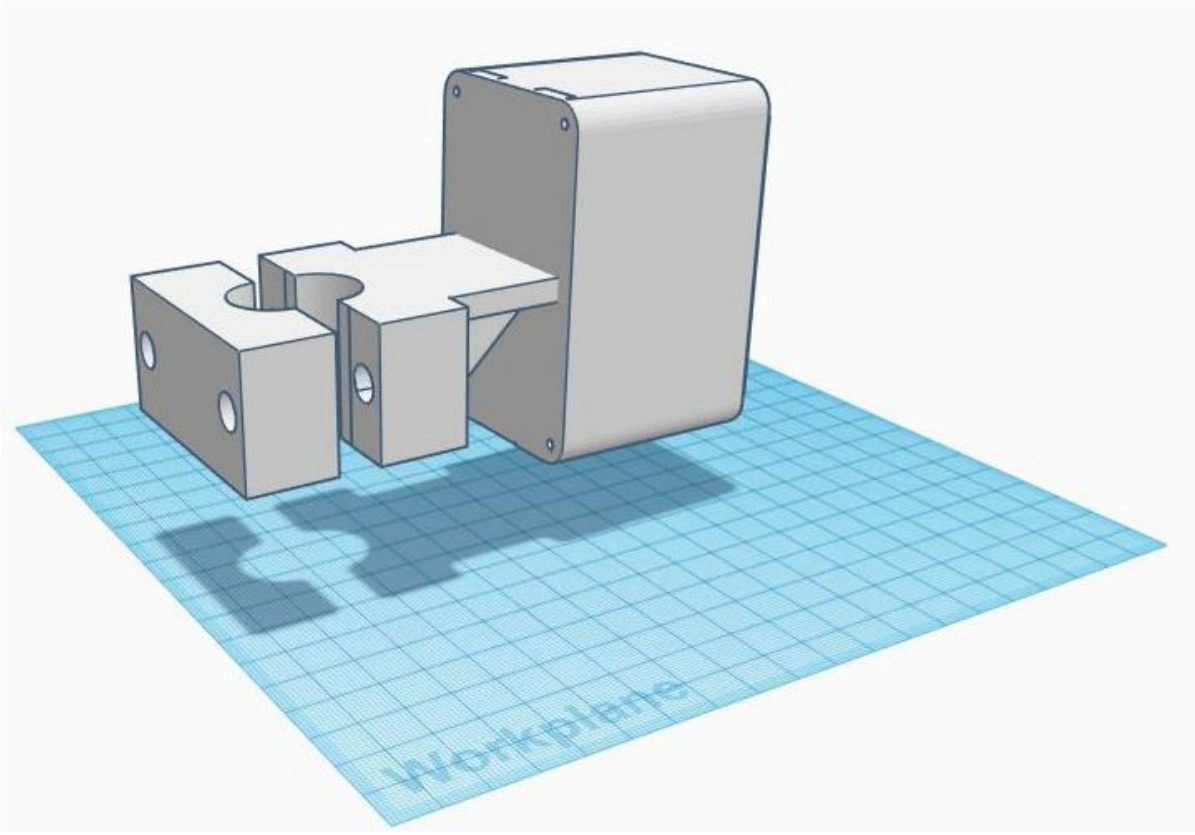
Για την τρισδιάστατη εκτύπωση των εξαρτημάτων βασική πηγή αποτέλεσε το project «Solar Weather Station Temperature/Humidity/Pressure (http://www.thingiverse.com/thing:2253801) by Skulbl4k4», καθώς επίσης έγινε και χρήση των λογισμικών σχεδίασης fusion 360 και tinkercad της Autodesk.

- Αδιάβροχο περίβλημα του αισθητήρα BME280 μαζί με βάση στήριξης.



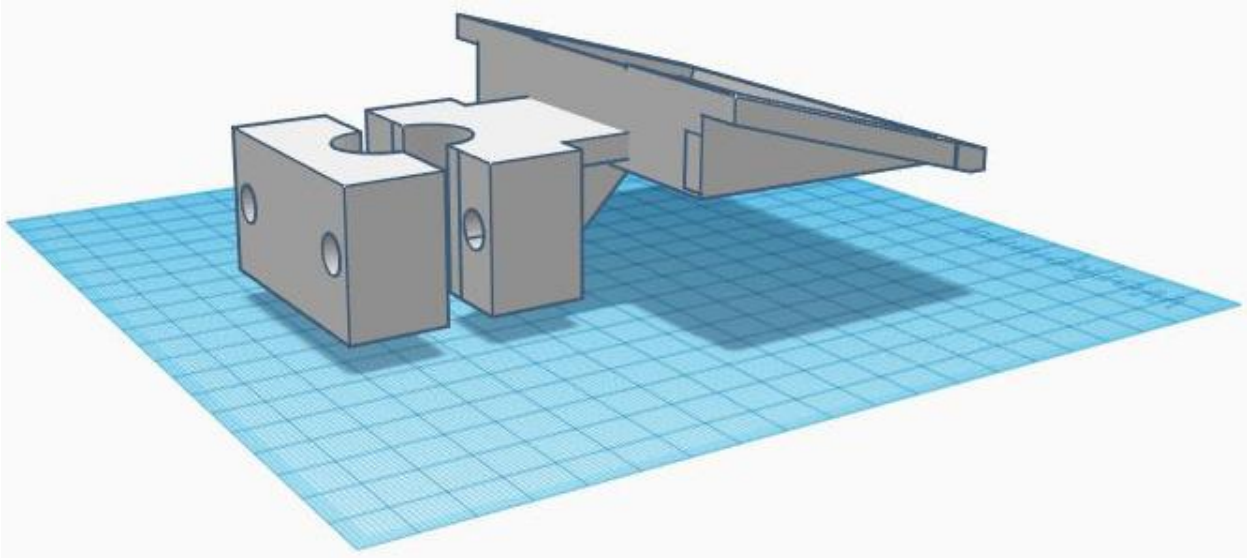
Εικόνα 31: Τρισδιάστατο σχέδιο περιβλήματος BME 280

- Βάση στήριξης ηλεκτρολογικού κουτιού που περιέχει την πλακέτα Sodaq ExpLoRer μαζί με την εξωτερική μπαταρία.

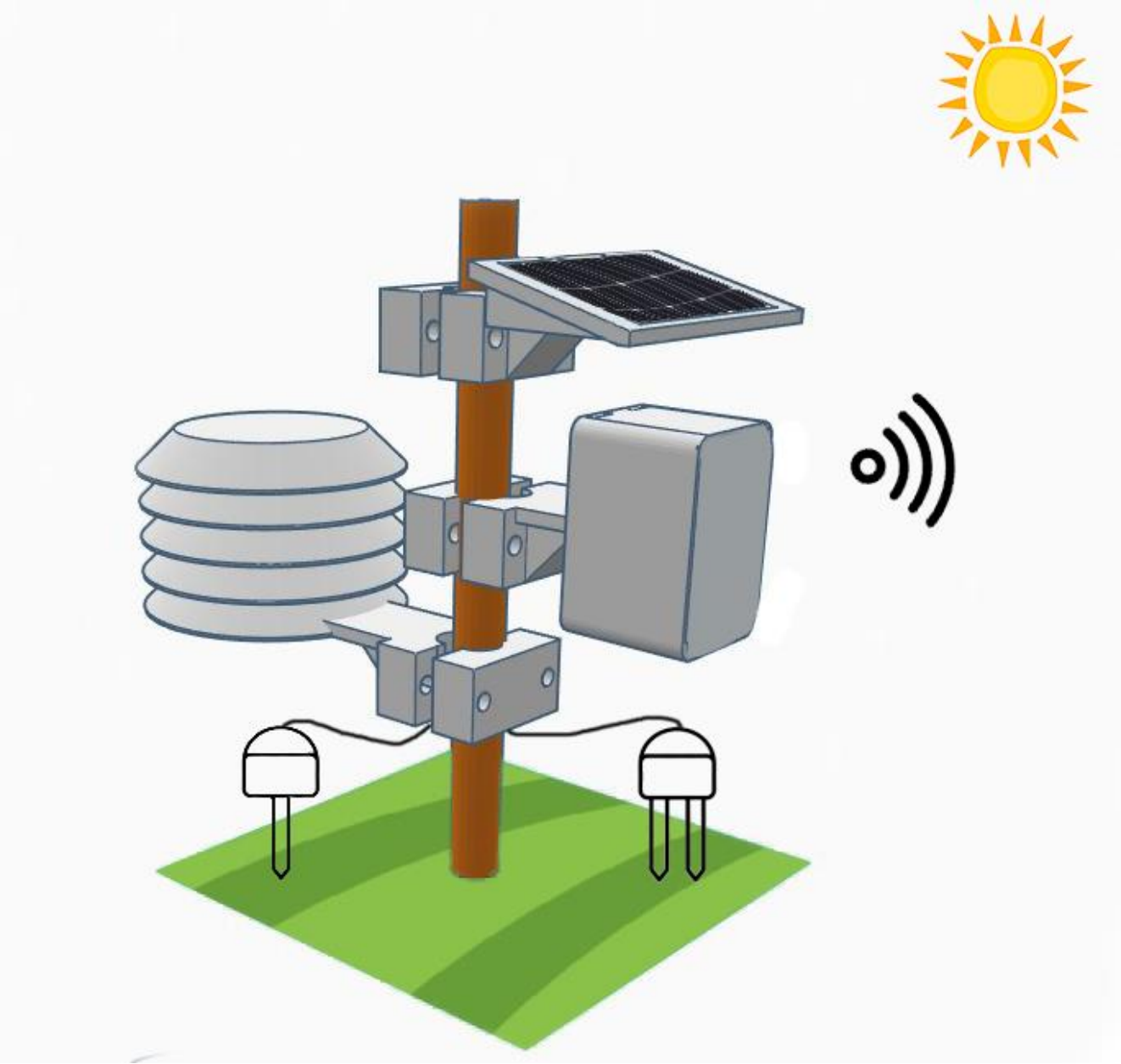


Εικόνα 32: Τρισδιάστατο σχέδιο βάσης στήριξης ηλεκτρολογικού κουτιού

- Βάση στήριξης ηλιακού πάνελ.



Εικόνα 33: Τρισδιάστατο σχέδιο βάσης ηλιακού πάνελ



Εικόνα 34: Τρισδιάστατο σχέδιο απεικόνισης αισθητήριου κόμβου

Μετά από την τρισδιάστατη εκτύπωση όλων τον απαραίτητων εξαρτημάτων, οι αισθητήρες και η πλακέτα του μικροελεγκτή τοποθετήθηκαν σε μεταλλική δοκό με τέτοιο τρόπο ώστε ο κόμβος να έχει κατεύθυνση προς τον σταθμό βάσης και το ηλιακό πάνελ να έχει όσο το δυνατόν καλύτερη ηλιακή κάλυψη με προσανατολισμό στο νότο.

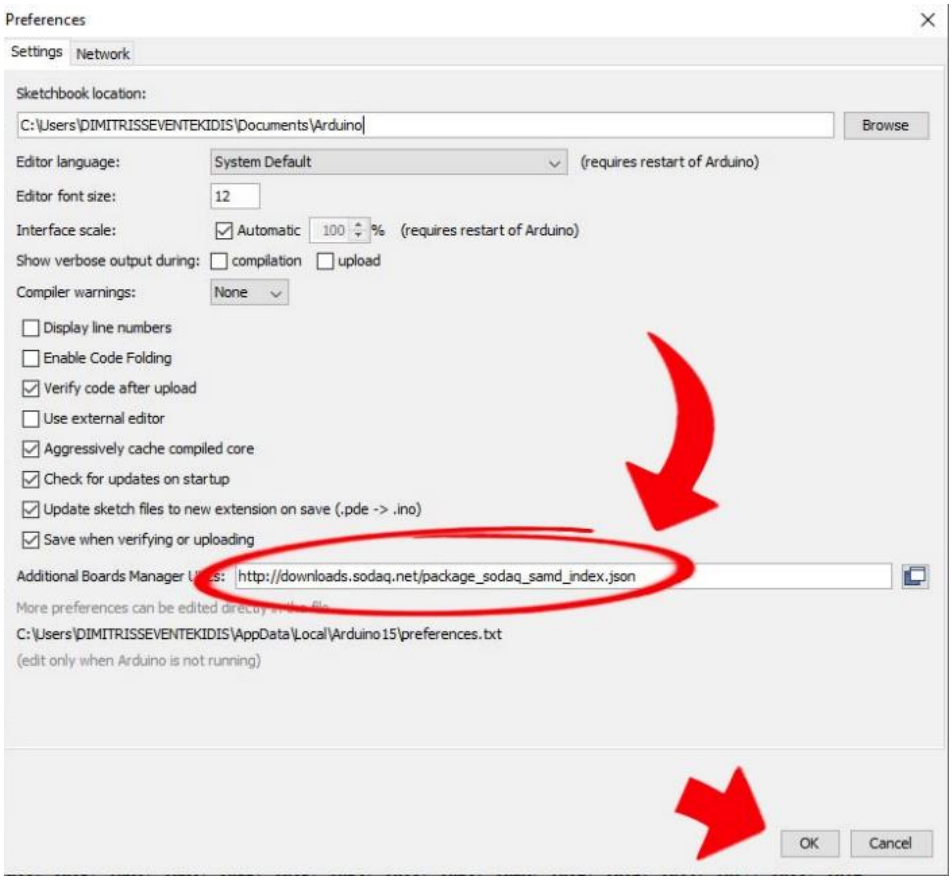
2.5 Προγραμματισμός κόμβου

Ακολουθεί βήμα προς βήμα η διαδικασία προγραμματισμού του μικροελεγκτή SODAQ ExpLoRer καθώς και όλων των συσκευών που αποτέλεσαν τον αισθητήριο κόμβο. Επίσης θα περιγραφεί και η διαδικασία αποστολής μετρήσεων στο The Things Network.

2.5.1 Προετοιμασία IDE

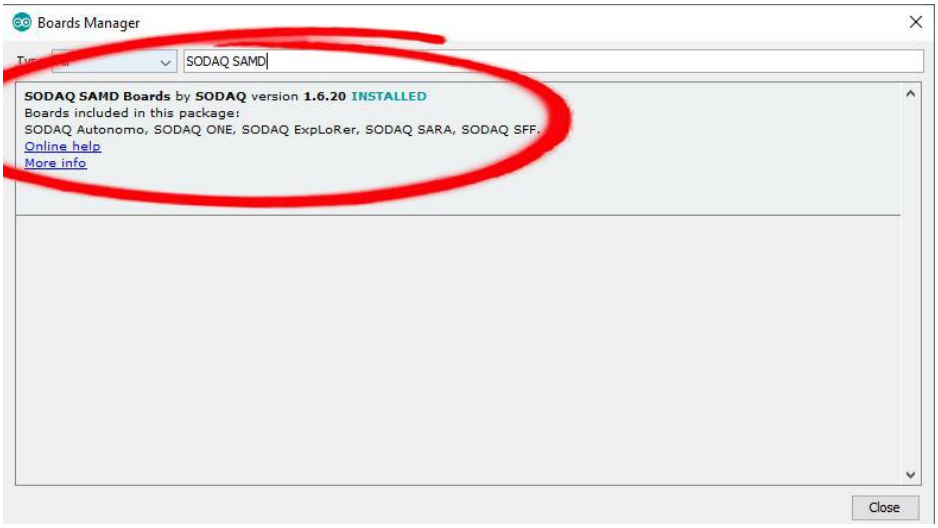
Η διαδικασία που απαιτείται για τον προγραμματισμό ενός SODAQ ExpLoRer [29] είναι παρόμοια με αυτή ενός οποιουδήποτε μικροελεγκτή Arduino , επομένως γίνεται μέσω του περιβάλλοντος προγραμματισμού Arduino IDE σε γλώσσα που βασίζεται στη C/C++. Η μόνη ενέργεια που απαιτείται από τον προγραμματιστή είναι να προσθέσει την πλακέτα SODAQ ExpLoRer στο περιβάλλον εργασίας του Arduino IDE , ακολουθώντας την παρακάτω διαδικασία.

- Αρχικά απαιτείται η φόρτωση ενός προσαρμοσμένου αρχείου πλακέτας ειδικά σχεδιασμένο για το SODAQ ExpLoRer. Η URL διεύθυνση του αρχείο είναι η : http://downloads.sodaq.net/package_sodaq_samd_index.json
  και για να το φορτώσει ο προγραμματιστής , θα πρέπει να ανοίξει το παράθυρο των προτιμήσεων «Αρχείο → Προτιμήσεις» και να επικολλήσει στο πεδίο «Πρόσθετη διαχείριση διευθύνσεων πλακέτας» το προηγούμενο URL.



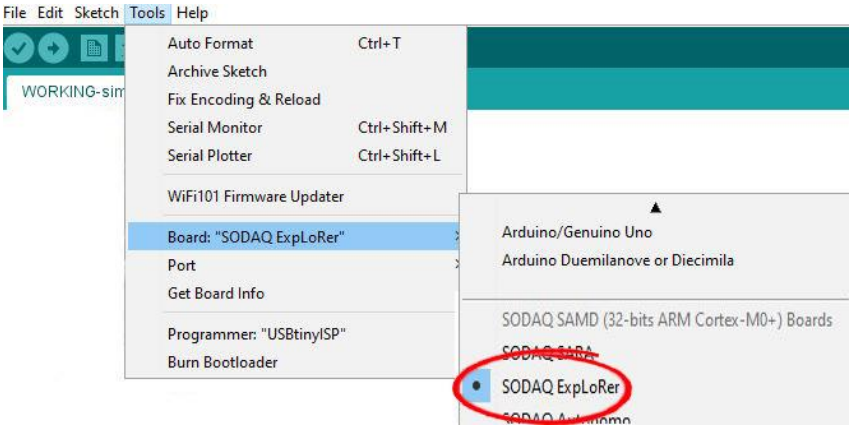
Εικόνα 35: Προσαρμοσμένο αρχείο SODAQ

- Έπειτα απαιτείται η εγκατάσταση της πιο πρόσφατης έκδοσης των μικροελεγκτών SODAQ SAMD , από τη διαδρομή «Εργαλεία → Πλακέτα → Διαχείριση πλακέτας».

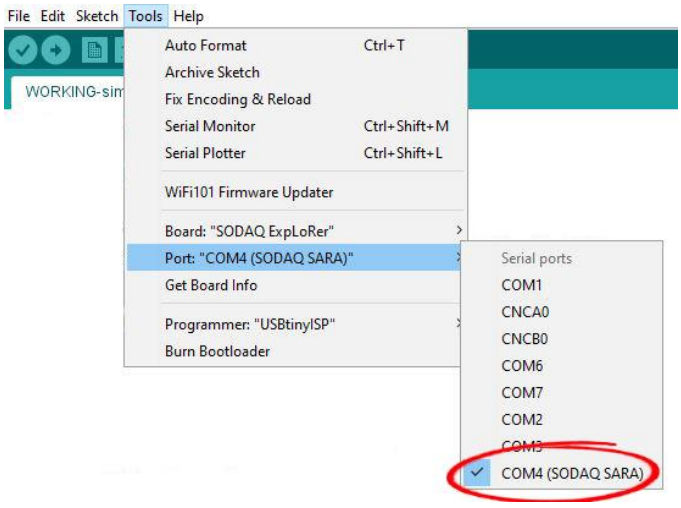


Εικόνα 36: Εγκατάσταση πρόσφατων SODAQ SAMD

- Τέλος απαιτείται η διαμόρφωση του υλικού επιλέγοντας την κατάλληλη πλακέτα , από τη διαδρομή «Εργαλεία → Πλακέτα → SODAQ ExpLoRer» καθώς και επιλογή της κατάλληλης COM θύρας από τη διαδρομή «Εργαλεία → Θύρα → (κατάλληλη θύρα)».



Εικόνα 37: Επιλογή πλακέτας



Εικόνα 38: Επιλογή θήρας

Με την ολοκλήρωση της παραπάνω διαδικασίας το SODAQ ExpLoRer [30] είναι έτοιμο για προγραμματισμό. Χρειάζονται επιπλέον βιβλιοθήκες για τη χρήση της εκάστοτε λειτουργίας , όπως για παράδειγμα τη σύνδεση μέσω LoRa και την ανάγνωση αισθητήρων, οι οποίες θα εξηγηθούν παρακάτω στην ανάλυση του κώδικα.

2.5.2 Κώδικας Arduino

Όπως έχει ήδη αναφερθεί , ο προγραμματισμός του SODAQ ExpLoRer δεν διαφέρει από τον προγραμματισμό οποιουδήποτε μικροελεγκτή Arduino. Αρχικά γίνεται η συμπερίληψη όλων των απαραίτητων βιβλιοθηκών και η δήλωση των μεταβλητών, έπειτα ο κώδικας χωρίζεται σε δύο κύριες συναρτήσεις , την **void setup()** , η οποία τρέχει μόνο μια φορά κατά την έναρξη της συσκευής στην οποία γίνεται αρχικοποίηση των απαραίτητων παραμέτρων , και την **void loop()** η οποία εκτελείται συνεχώς και περιλαμβάνει όλες τις διεργασίες για τη συνεχή λειτουργεία του αισθητήριου κόμβου. Έπειτα ακολουθούν οι βοηθητικές συναρτήσεις που καλούνται στην void loop().

2.5.2.1 Αρχικοποίηση του προγράμματος

Αρχικά γίνεται συμπερίληψη όλων των απαραίτητων βιβλιοθηκών για την ομαλή λειτουργία του κώδικα:

```
#include <Sodaq_RN2483.h>
```

- Η Βιβλιοθήκη Sodaq_RN2483.h σχεδιασμένη ειδικά για το Microchip RN2483 υποστηρίζει την αμφίδρομη ανταλλαγή μηνυμάτων συσκευών Class A και των σταθμών βάσης , με το πρωτόκολλο LoRaWAN.

```
#include <OneWire.h>
#include <DallasTemperature.h>
```

- Η βιβλιοθήκη OneWire.h επιτρέπει την πρόσβαση σε συσκευές που βασίζονται στο σύστημα 1-Wire (ενός καλωδίου), το οποίο είναι ένα σύστημα δίαυλου επικοινωνίας συσκευών σχεδιασμένο για την μετάδοση ψηφιακών σημάτων με τη χρήση ενός αγωγού. Στη συγκεκριμένη εργασία , ο συνδυασμός OneWire.h με

την DallasTemperature.h ήταν απαραίτητος για τη λήψη μετρήσεων του αισθητήρα θερμοκρασίας DS18b20.

```
#include <LiquidCrystal_I2C.h>
```

- Η βιβλιοθήκη LiquidCrystal_I2C.h επιτρέπει το χειρισμό οθόνης LCD μέσω του σειριακού συστήματος διαύλου επικοινωνίας I2c , το οποίο είναι παρόμοιο με το 1-Wire.

```
#include <Wire.h>
```

- Η Wire.h είναι και αυτή μια βιβλιοθήκη που επιτρέπει την επικοινωνία μιας συσκευής και μια πλακέτας Arduino μέσω I2c.

```
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

- Η βιβλιοθήκη SPI.h επιτρέπει την επικοινωνία συσκευών μέσω του πρωτοκόλλου SPI. Το SPI (Serial Peripheral Interface) είναι ένα σύγχρονο πρωτόκολλο ανταλλαγής σειριακών δεδομένων σε μικρές αποστάσεις. Στη συγκεκριμένη εργασία τα δεδομένα του αισθητήρα BME280 λαμβάνονται μέσω των βιβλιοθηκών Adafruit_Sensor.h [31] και Adafruit_BME280.h και στη συνέχεια αποστέλλονται μέσω SPI , όπου Master συσκευή είναι ο μικροελεγκτής SODAQ και Slave ο αισθητήρας.

Έπειτα ακολουθεί ο καθορισμός σταθερών τιμών και η ονομασία μεταβλητών με καθολική ισχύ (Global variables):

```
#define debugSerial SerialUSB
```

- Η αναφορά στο όνομα «SerialUSB» αντικαθίσταται από το «debugSerial» , καθ' όλη την εκτέλεση του κυρίως προγράμματος.

```
#define USEIIC 0
#if(USEIIC)
  Adafruit_BME280 bme;
#else
  #define SPI_SCK 13
  #define SPI_MISO 12
  #define SPI_MOSI 11
  #define SPI_CS 10
  Adafruit_BME280 bme(SPI_CS, SPI_MOSI, SPI_MISO, SPI_SCK);
#endif
```

- Καθώς ο αισθητήρας BME 280 υποστηρίζει τόσο I2C όσο και SPI , μέσω της χρήσης της USEIIC δίνετε η δυνατότητα να εναλλάσσεται άμεσα ο τρόπος επικοινωνίας, ορίζοντας την USEIIC σε 0 για SPI και 1 για i2c (προϋποτίθεται και η αλλαγή της συνδεσμολογίας). Έπειτα ακολουθεί συνθήκη που εξετάζει τον τρόπο επικοινωνίας που έχει οριστεί και ορίζει αντίστοιχη συνδεσμολογία. Η

παραπάνω διαδικασία θα μπορούσε να είχε παραληφθεί αλλά αποτελεί εύκολη και γρήγορη λύση σε περίπτωση επαναχρησιμοποίησης του κώδικα για χρήση με άλλη συνδεσμολογία.

```
#define loraSerial Serial2
```

- Η αναφορά στο όνομα «Serial2» , που αποτελεί την σειριακή επικοινωνία της πλακέτας LoRa αντικαθίσταται από το «loraSerial» , καθ' όλη την εκτέλεση του κυρίως προγράμματος.

```
#define NIBBLE_TO_HEX_CHAR(i) ((i <= 9) ? ('0' + i) : ('A' - 10 + i))
#define HIGH_NIBBLE(i) ((i >> 4) & 0x0F)
#define LOW_NIBBLE(i) (i & 0x0F)
```

- Οι παραπάνω εντολές καθορισμού των NIBBLE_TO_HEX_CHAR, HIGH_NIBBLE και HIGH_NIBBLE είναι απαραίτητες για την εμφάνιση του μοναδικού HWEUI αριθμού της συσκευής σε κατάλληλη μορφή.

```
#define ONE_WIRE_BUS 8
```

- Γίνεται ονομασία της ψηφιακής εισόδου D8 σε ONE_WIRE_BUS , η συγκεκριμένη είσοδος χρησιμοποιείται για την λήψη των δεδομένων του αισθητήρα DS18b20.

```
unsigned long delayTime;
```

- Ορίζεται μεταβλητή τύπου unsigned long με όνομα delayTime , η οποία αρχικοποιείται μετέπειτα στην συνάρτηση setup(). Ο ρόλος της συγκεκριμένης μεταβλητής στο κυρίως πρόγραμμα είναι να καθορίζει τον χρόνο ανάμεσα στις μετρήσεις.

```
int rainPin = A1;
```

- Ορίζεται μεταβλητή τύπου int με όνομα rainPin, η οποία αντιστοιχεί στην αναλογική είσοδο A1. Στην συγκεκριμένη μεταβλητή αποθηκεύεται η μέτρηση του αισθητήρα υγρασίας εδάφους.

```
float vout = 0.0;
float vin = 0.0;
int value = 0;
```

- Ορίζονται οι μεταβλητές τύπου float vout, vin και value και αρχικοποιούνται για χρήση στην μέτρησης της τάσης του κυκλώματος τροφοδοσίας.

```
bool OTAA = true;
```

- Ορίζεται μεταβλητή τύπου bool με όνομα OTAA και αρχικοποίηση true , η οποία ορίζει τον τρόπο με τον οποίο η συσκευή θα συνδεθεί στο δίκτυο LoRaWAN. Στην περίπτωση που η μεταβλητή OTAA είναι true η εγκαθίδρυση επικοινωνίας γίνεται με OTAA δηλαδή με Over the air activation , ενώ σε περίπτωση που είναι false , η επικοινωνία επιτυγχάνεται με ABP (Activation by personalization) . Η συγκεκριμένη μεταβλητή δεν είναι απαραίτητη αλλά χρησιμεύει στο να επαναχρησιμοποιηθεί ο ίδιος κώδικας με εύκολα εναλλαγή σε ABP.

```
const uint8_t devAddr[] = {};
const uint8_t appSKey[] = {};
const uint8_t nwkSKey[16] = {};
```

- Ορίζονται σταθερές πινάκων τύπου uint8_t οι οποίες περιέχουν τα διαπιστευτήρια σε περίπτωση ABP σύνδεσης , στους συγκεκριμένους πίνακες

δεν εκχωρείται κάποια τιμή γιατί στην περίπτωση μας χρησιμοποιείται OTAA. Όπως και στην περίπτωση της προηγούμενης μεταβλητής τα παραπάνω κομμάτια κώδικα θα μπορούσαν να έχουν παραληφθεί.

```
static uint8_t DevEUI[8] ={ 0x00, 0x04, 0xA3, 0x0B, 0x00, 0x1F, 0xCC, 0x27 };
```

- Ορίζεται μεταβλητή static τύπου uint8_t σε μορφή πίνακα (οκτώ θέσεων), η οποία περιέχει τον μοναδικό αριθμό HWEUI του SODAQ.

```
const uint8_t AppEUI[8] = { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x03, 0x91, 0x1E };
```

- Ορίζεται σταθερά τύπου uint8_t σε μορφή πίνακα (οκτώ θέσεων) με όνομα DevEUI, η οποία περιέχει τον μοναδικό αριθμό Application EUI του The Things Network application με τη συσκευή μας.

```
const uint8_t AppKey[16] = { 0x63, 0x0D, 0xA6, 0xA4, 0x1B, 0x6D, 0xD4, 0x03,
0xBC, 0xBA, 0x3E, 0x82, 0x64, 0x97, 0x85, 0x6E };
```

- Ορίζεται σταθερά τύπου uint8_t σε μορφή πίνακα (δεκαέξι θέσεων) με όνομα AppKey, η οποία περιέχει τον μοναδικό αριθμό App Key του The Things Network application με τη συσκευή μας. Οι παραπάνω πίνακες περιέχουν τα διαπιστευτήρια της εφαρμογής και μπορούν να δώσουν πρόσβαση σε τρίτους , επομένως έγκειται προσοχή ως προς την κοινοποίησή τους.

```
OneWire oneWire(ONE_WIRE_BUS);
```

- Γίνεται χρήση της OneWire.

```
Dallas Temperature sensors(&oneWire);
```

- Γίνεται χρήση της DallasTemperature.

```
float dallas_Celcius=0;
```

- Ορίζεται μεταβλητή τύπου float με όνομα dallas_Celcius και αρχικοποιείται με την τιμή 0 , στη συγκεκριμένη μεταβλητή χειρίζεται δεδομένα του DS18b20.

2.5.2.2 Συνάρτηση void setup

```
void setup(void)
{

  debugSerial.begin(9600);
```

- Γίνεται η έναρξη της void setup και καλείται η debugSerial.begin με ρυθμό μετάδοσης (baud rate) 9600 η οποία εκκινεί το serial monitor. Η χρήση του serial monitor ήταν εξαιρετικά σημαντική στην υλοποίηση του προγράμματος χωρίς σφάλματα ,καθώς ήταν εφικτό να γίνεται έλεγχος σε πραγματικό χρόνο.

```
pinMode(rainPin, INPUT);
```

- Ορίζεται ως είσοδος το pin rainPin (αντιστοιχίστηκε με το A1).

```
sensors.begin();
 bool rslt;
    rslt = bme.begin();
    if (!rslt) {
```

```
        debugSerial.println("BME failed to initialize , check wirring");
        while (1); }
    debugSerial.println("Init Success");
    debugSerial.println("Temperature      Pressure       Humidity");
```

- Αρχικοποιείται ο αισθητήρας DS18b20 , καθώς και ο BME280. Έπειτα εμφανίζεται στην οθόνη κατάλληλο μήνυμα αν έχει επιτευχθεί η αρχικοποίηση του BME280 , τέλος εκτυπώνεται το μήνυμα «Temperature Pressure Humidity» , όπου κατά την κλήση της αντίστοιχης συνάρτησης θα εμφανίζονται από κάτω οι αντίστοιχες τιμές.

```
 delayTime = 1000;
 debugSerial.println("Start");
 loraSerial.begin(LoRaBee.getDefaultBaudRate());
 LoRaBee.setDiag(debugSerial);
 LoRaBee.init(loraSerial, LORA_RESET);
 getHWEUI();
 debugSerial.print("LoRa HWEUI: ");

  for (uint8_t i = 0; i < sizeof(DevEUI); i++) {
      debugSerial.print((char)NIBBLE_TO_HEX_CHAR(HIGH_NIBBLE(DevEUI[i])));
      debugSerial.print((char)NIBBLE_TO_HEX_CHAR(LOW_NIBBLE(DevEUI[i])));
  }
  debugSerial.println();

  setupLoRa();
```

- Αρχικοποιείται η delayTime και έπειτα γίνεται ρύθμιση του LoRa με χρήση του κωδικού HWEUI της συσκευής και εμφανίζονται στην οθόνη κατάλληλα μηνύματα.

```
lcd.init();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(1,0);
  lcd.print("Sodaq LoRa node");
  lcd.setCursor(1,1);
  lcd.print("Services up"); }
```

- Γίνεται αρχικοποίηση της οθόνης lcd , ενεργοποιείται το backlight της και έπειτα τοποθετείται ο κέρσορας σε κατάλληλες θέσεις και εμφανίζονται μηνύματα ενεργοποίησης της συσκευής. Η κλήση της lcd.setCursor με ορίσματα 1, 0 σημαίνει πως ο κέρσορας τοποθετείται στην πρώτη θέση (1 από 16) της πρώτης γραμμής (0 ή 1). Τέλος κλείνει η συνάρτηση void main.

2.5.2.3 Συνάρτηση void loop

```
void loop(void)
{

  bme_values();
  dallas_temperature();
  moisture_sensor();
```

- Αρχίζει η συνάρτηση void loop , έπειτα καλούνται οι συναρτήσεις bme_values , dallas_temperature και moisture_sensor , οι οποίες κατά την κλήση τους εμφανίζουν στην οθόνη τις πιο πρόσφατες μετρήσεις των αισθητήρων.

```
String reading = getValues();
debugSerial.println(reading);
```

- Καλείται η συνάρτηση getValues , της οποίας η επιστροφή αποθηκεύεται στη μεταβλητή reading τύπου String. Η μεταβλητή reading είναι από τις πιο σημαντικές σε όλο το πρόγραμμα καθώς περιέχει το τελικό μήνυμα που θα στείλει μέσω LoRa η συσκευή μας.

```
   switch (LoRaBee.send(1, (uint8_t*)reading.c_str(), reading.length()))
    {
    case NoError:
      debugSerial.println("Successful transmission.");
      break;
    case NoResponse:
      debugSerial.println("There was no response from the device.");
      break;
    case Timeout:
      debugSerial.println("Connection timed-out. Check your serial connection
to the device! Sleeping for 20sec.");
      delay(20000);
      break;
    case PayloadSizeError:
      debugSerial.println("The size of the payload is greater than allowed.
Transmission failed!");
      break;
    case InternalError:
      debugSerial.println("Oh No! This shouldn't happen. Something is really
wrong! The program will reset the RN module.");
      setupLoRa();
      break;
    case Busy:
      debugSerial.println("The device is busy. Sleeping for 10 extra
seconds.");
      delay(10000);
      break;
    case NetworkFatalError:
      debugSerial.println("There is a non-recoverable error with the network
connection. The program will reset the RN module.");
      setupLoRa();
      break;
    case NotConnected:
      debugSerial.println("The device is not connected to the network. The
program will reset the RN module.");
      setupLoRa();
      break;
    case NoAcknowledgment:
      debugSerial.println("There was no acknowledgment sent back!");
      break;
    default:
      break; }
```

- Γίνεται προσπάθεια αποστολής του μηνύματος reading στο δίκτυο μέσω LoRa , έπειτα γίνεται έλεγχος ροής με μια δομή ελέγχου switch () {} . Στην περίπτωση που η μετάδοση είναι επιτυχής εμφανίζεται μήνυμα επιτυχίας, ενώ σε διαφορετική περίπτωση εμφανίζεται το κατάλληλο μήνυμα.

Η εμφάνιση των μετρήσεων στην οθόνη lcd γίνεται με το παρακάτω κομμάτι κώδικα.

```
lcd.init();
lcd.init();
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("Air Sensors");
lcd.setCursor(0,1);
lcd.print("       --->");
delay(delayTime);
lcd.init();
lcd.init();
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("Temp:");
lcd.print (bme.readTemperature());
lcd.print((char)223);
lcd.print("C");
lcd.setCursor(0,1);
lcd.print(" Hum:");
lcd.print (bme.readHumidity());;
lcd.print("%");
delay(delayTime);
lcd.init();
lcd.init();
lcd.setCursor(0,0);
lcd.print("Press:");
float pressure = (bme.readPressure()/100.0F) ;
lcd.print (pressure);
lcd.print("hPa");
```

- Αρχικά χρησιμοποιείται η εντολή αρχικοποίησης lcd.init(); για να διαμορφωθεί η lcd οθόνη , έπειτα ενεργοποιείται ο φωτισμός backlight και τοποθετείται ο κέρσορας. Έπειτα εμφανίζεται στην lcd μήνυμα που προετοιμάζει τον χρήστη για τις μετρήσεις των ατμοσφαιρικών αισθητήρων που θα ακολουθήσουν. Τέλος εμφανίζονται στην lcd τα δεδομένα ατμοσφαιρικής θερμοκρασίας , υγρασίας και πίεσης μέσω των bme.readTemperature() , bme.readHumidity() και bme.readPressure().

```
delay(delayTime);
lcd.init();
lcd.init();

lcd.setCursor(0,0);
lcd.print("Ground Sensors");
lcd.setCursor(0,1);
lcd.print("          --->");

lcd.init();
lcd.init();
```

```
  lcd.setCursor(0,0);
  lcd.print("G_Temp:");
  lcd.print (sensors.getTempCByIndex(0));
  lcd.print((char)223);
  lcd.print("C");

  float grmois = analogRead(rainPin);
  grmois = 100 - (((grmois-260) / (1023-260)) * 100);
  lcd.setCursor(0,1);
  lcd.print("G_Mois:");
  lcd.print (grmois);;
  lcd.print("%");
  delay(delayTime);
```

- Αφού μεσολαβήσει κάποιο χρονικό διάστημα μέσω της delay(delayTime); χρησιμοποιείται η εντολή αρχικοποίησης lcd.init(); για να διαμορφωθεί η lcd οθόνη , έπειτα τοποθετείται ο κέρσορας. Έπειτα εμφανίζεται στην lcd μήνυμα που προετοιμάζει τον χρήστη για τις μετρήσεις των αισθητήρων εδάφους που θα ακολουθήσουν. Έπειτα εμφανίζεται στην lcd η θερμοκρασία εδάφους. Όσον αφορά την υγρασία στο έδαφος , αρχικά ορίζεται μεταβλητή grmois τύπου float , στη οποία εκχωρείται η τιμή που επιστρέφει η analogRead(rainPin). Η τιμή που επιστρέφει η analogRead(rainPin) είναι ένας ακέραιος αριθμός από το 260 έως το 1023 , για να μετατραπεί σε ένα ποσοστό επί τις εκατό , εφαρμόζεται η συνάρτηση [Ποσοστό υγρασίας={ 100-((grmois-ελάχιστη_τιμή) ÷(μέγιστη_τιμή - ελάχιστη_τιμή)}]. Τέλος εμφανίζεται στην lcd το ποσοστό της υγρασίας εδάφους επί τοις εκατό.

2.5.2.4 Συναρτήσεις

```
String getValues()
{
  float airtemp;
  float airhum;
  float airpres;
  float grtemp;
  float grmois;
  float solar;

  String total;

  String airtemp1;
  String airhum1;
  String airpres1;
  String grtemp1;
  String grmois1;
  String solar1;

  airtemp = (bme.readTemperature());
  airhum = (bme.readHumidity());
```

```
  airpres = (bme.readPressure()/100.0F);
  sensors.requestTemperatures();
  dallas_Celcius=sensors.getTempCByIndex(0);
  grtemp = dallas_Celcius;
  grmois = analogRead(rainPin);
  grmois = 100 - (((grmois-260) / (1023-260)) * 100);

  airtemp1 = airtemp;
  airhum1 = airhum;
  airpres1 = airpres;
  grtemp1 = grtemp;
  grmois1 = grmois;
  value = analogRead(voltageSensor);
  vout = (value * 3.3) / 1024.0;
  vin = vout / (7500.0/(30000.0+7500.0));
  solar = vin;

  total = airtemp1 + airhum1 + airpres1 + grtemp1 + grmois1 + solar;

  return String(total);}
```

- Η συνάρτηση getValues() αποτελεί ένα από τα πιο σημαντικά κομμάτια του προγράμματος καθώς είναι αυτή που συνδυάζει τις τιμές όλων των αισθητήρων και τελικά επιστρέφει μια μεταβλητή τύπου String η οποία περιέχει όλα τα δεδομένα και στη συνέχεια αποστέλλεται μέσω LoRa στο δίκτυο. Πιο συγκεκριμένα , αρχικά ορίζονται μεταβλητές τύπου float οι οποίες χειρίζονται σε πρώτη φάση τα δεδομένα των αισθητήρων , επίσης ορίζονται , η μεταβλητή total και βοηθητικές μεταβλητές τύπου String για τον χειρισμό των δεδομένων στην τελική φάση. Έπειτα καλούνται οι κατάλληλες συναρτήσεις και αποθηκεύουν τα δεδομένα στις μεταβλητές float. Τέλος τα δεδομένα μετατρέπονται σε String και συγχωνεύονται στην μεταβλητή total την οποία και επιστρέφει η συνάρτηση.

```
static void getHWEUI()
{ uint8_t len = LoRaBee.getHWEUI(DevEUI, sizeof(DevEUI)); }
```

- Η συνάρτηση getHWEUI() επιστρέφει τον μοναδικό αριθμό HWEUI.

```
void setupLoRa(){
  if(!OTAA){ setupLoRaABP(); }
  else {
   setupLoRaOTAA();
  }
  LoRaBee.setSpreadingFactor(9); }
```

- Η συνάρτηση setupLoRa() ελέγχει αν η ενεργοποίηση έχει οριστεί με OTAA ή με ABP , και καλεί την αντίστοιχη συνάρτηση.

```
void setupLoRaOTAA(){

  if (LoRaBee.initOTA(loraSerial, DevEUI, AppEUI, AppKey, true))
  {
        debugSerial.println("Network connection successful."); }
  else {
    debugSerial.println("Network connection failed!"); } }
```

- Η συνάρτηση setupLoRaOTAA () ελέγχει αν η ενεργοποίηση της συσκευής με OTAA ήταν επιτυχής και εμφανίζει κατάλληλο μήνυμα.

```
void dallas_temperature(){
  sensors.requestTemperatures();
  dallas_Celcius=sensors.getTempCByIndex(0);
  debugSerial.print("Ground Temp is(C): ");
  debugSerial.println(dallas_Celcius);  }
```

- Η συνάρτηση dallas_temperature() διαβάζει την θερμοκρασία εδάφους από τον αισθητήρα DS18b20 και την εμφανίζει στο serial monitor.

```
void bme_values() {
    debugSerial.print("temperature:");
    debugSerial.print(bme.readTemperature());
    debugSerial.print("*C   ");

    debugSerial.print("pressure:");
    debugSerial.print(bme.readPressure()/100.0F);
    debugSerial.print("hPa   ");

    debugSerial.print("humidity:");
    debugSerial.print(bme.readHumidity());
    debugSerial.println("%   ");   }
```

- Η συνάρτηση bme_values() διαβάζει τις τιμές ατμοσφαιρικής θερμοκρασίας , υγρασίας και πίεσης του αισθητήρα BME280 και τέλος τις εμφανίζει στο serial monitor.

```
void moisture_sensor(){
  float sensorValue1 = analogRead(rainPin);
  float sensorValue = 100 - (((sensorValue1-260) / (1023-260)) * 100);
  debugSerial.print("Yl raw value:");
  debugSerial.println(sensorValue1);
  debugSerial.print("Yl percentage value:");
  debugSerial.println(sensorValue); }
```

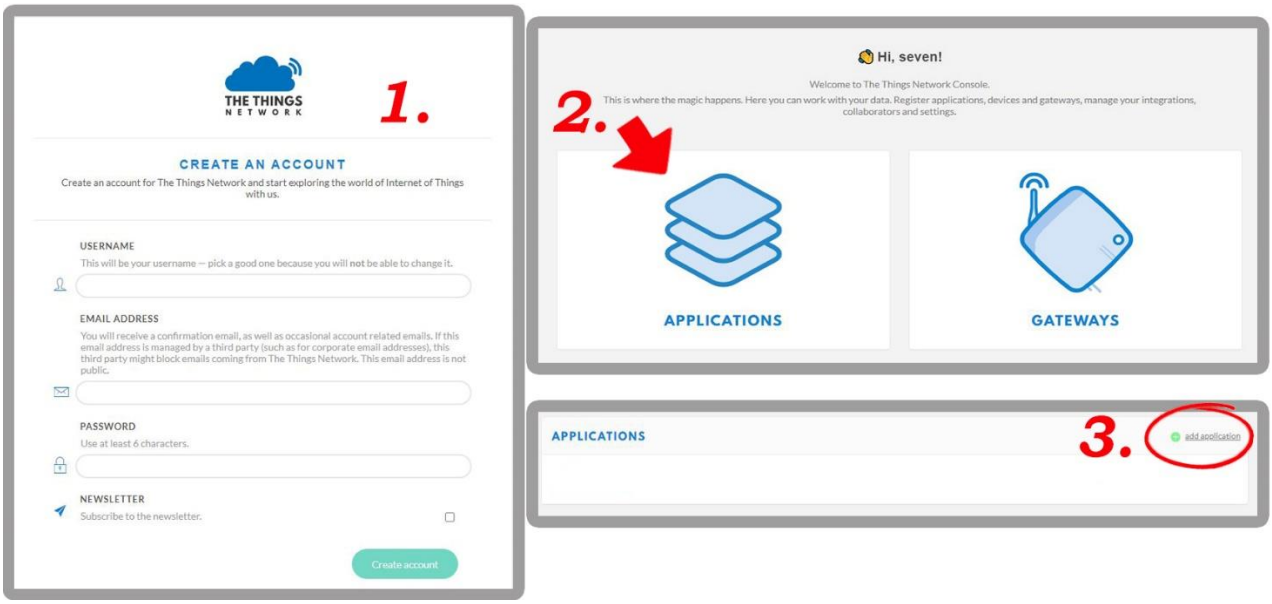
- Η συνάρτηση moisture_sensor() διαβάζει την αναλογική τιμή που επιστρέφει ο αισθητήρας YL-39 και αφού υπολογίσει την τιμή της υγρασίας σε ποσοστό την εμφανίζει στο serial monitor.

2.6 Δημιουργία application στο TTN

Παράλληλα με τον προγραμματισμό του κόμβου υλοποιήθηκε κατάλληλο application στο The things Network για τον χειρισμό των δεδομένων σε πρώτο στάδιο. Η συγκεκριμένη διαδικασία είναι προ απαιτούμενη για την προώθηση των δεδομένων σε άλλες υπηρεσίες , που είναι ενεργές σε κάποιο cloud ή κάποιον τοπικό sever.

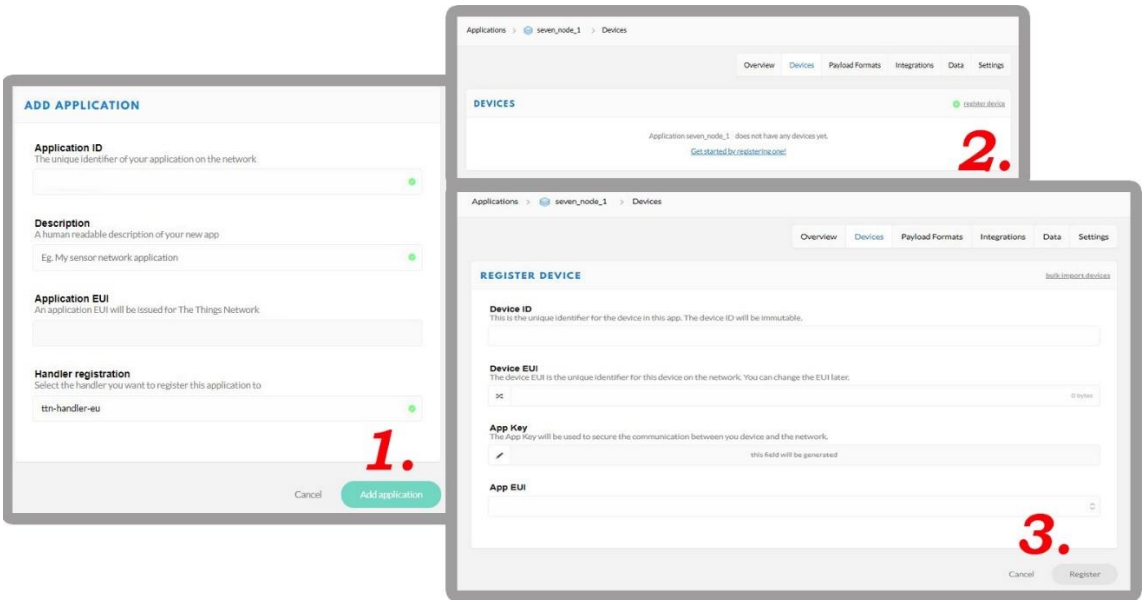
- Αρχικά έγινε εγγραφή στην πλατφόρμα του TTN, μέσω του συνδέσμου https://account.thethingsnetwork.org/register , και έπειτα μέσω της κονσόλας επιλέχθηκε η

προσθήκη καινούριου application.



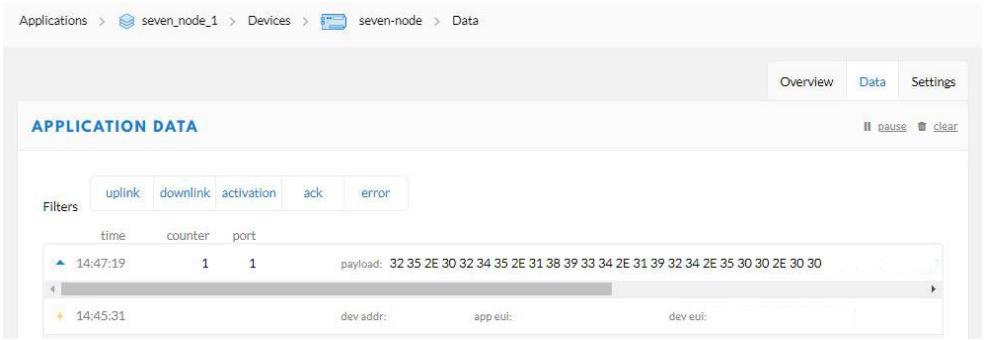
Εικόνα 39: Εγγραφή στην πλατφόρμα The Things Network

- Έπειτα δόθηκε ονομασία στο application με Application ID “seven_node_1”. Από την διαχείριση του application ακολούθησε η προσθήκη του αισθητήριου κόμβου LoRa που έχει ήδη προγραμματιστεί, στο πεδίο Device EUI αντιστοιχεί ο μοναδικός αριθμός αναγνώρισης HWEUI που έχει εξηγηθεί κατά την διαδικασία προγραμματισμού του Sodaq ExpLoRer. Το Device EUI καθώς και τα πεδία Application EUI και Application Key αποτελούν τα διαπιστευτήρια του κόμβου που προστέθηκαν κατά τον προγραμματισμό. Αξίζει να σημειωθεί πως η προκαθορισμένη τιμή του Handler registration είναι στη διεύθυνση ttn-handler-eu , αυτό είναι σημαντικό καθώς η συγκεκριμένη διεύθυνση αποτελεί τον MQTT broker της εφαρμογής μας από την οποία στη συνέχεια θα προωθήσουμε τα δεδομένα σε άλλες.



Εικόνα 40: Εγγραφή συσκευής στο TTN

- Μετά από την επιτυχημένη δημιουργία του application στο The Things Network , και τον σωστό προγραμματισμό του κόμβου ακολούθησε η άμεση επίβλεψη των δεδομένων από την αντίστοιχη επιλογή του TTN.



Εικόνα 41: Παρακολούθηση Payload

Το πρώτο μήνυμα που καταφθάνει είναι αυτό της ενεργοποίησης (Activation) μέσω OTAA , όπως έχει οριστεί στον προγραμματισμό του κόμβου. Περιλαμβάνει πληροφορίες «metadata» σχετικά με την επικοινωνία μέσω LoRaWAN, πιο συγκεκριμένα πληροφορίες τις ακριβούς ώρας μετάδοσης ,της διαμόρφωσης του σήματος LoRa καθώς και του gateway που έλαβε το σήμα.

- Το πεδίο time περιέχει τις πληροφορίες της ημερομηνίας και της ώρας.
- Στο πεδίο frequency αναγράφεται η συχνότητα μετάδοσης.
- Στο πεδίο modulation αναγράφεται η διαμόρφωση.
- Στο πεδίο data_rate αναγράφεται ο ρυθμός διασποράς.

- Το πεδίο coding_rate περιέχει τη ροή των δεδομένων που μεταφέρουν πραγματικές πληροφορίες.

  Το πεδίο Gateways , περιέχει υπό-πεδία που περιέχουν αναλυτικές πληροφορίες σχετικά με τα gateways που έλαβαν το σήμα.

  - Το πεδίο gtw_id περιέχει τον κωδικό αναγνώρισης του gateway.
  - Τα πεδία timestamp και time περιέχουν πληροφορίες σχετικά με την ακριβή ώρα που το gateway αντάλλαξε πληροφορίες με τον server και χρησιμεύουν στην περίπτωση πολλαπλών λήψεων από διαφορετικά gateways.
  - Το πεδίο channel περιέχει το κανάλι μετάδοσης.
  - Το πεδίο rssi περιέχει την ένδειξη στάθμης του ληφθέντος σήματος.
  - Το πεδίο snr περιέχει την πληροφορία του λόγου σήματος προς το θόρυβο.


```
{
  "time": "2021-01-05T12:45:30.177052363Z",
  "frequency": 868.1,
  "modulation": "LORA",
  "data_rate": "SF  BW   ",
  "coding_rate": "4/5",
  "gateways": [
    {
      "gtw_id": "                    ",
      "timestamp": 781345876,
      "time": "",
      "channel": 0,
      "rssi": -77,
      "snr": 8.5
    }
  ]
}
```


Εικόνα 42: Δεδομένα μηνύματος ενεργοποίησης

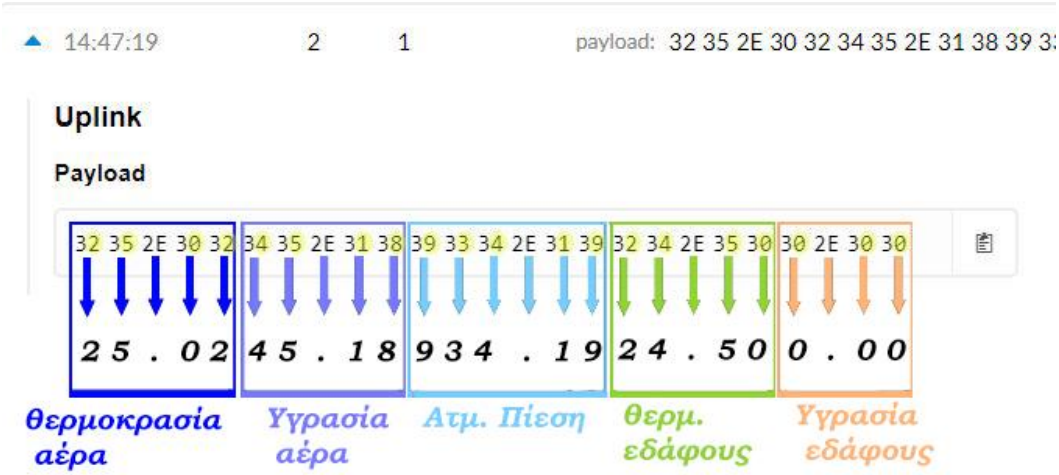
Όσον αφορά το επόμενο μήνυμα , πρόκειται για το πρώτο uplink μήνυμα από στάλθηκε από τον αισθητήριο κόμβο , εκτός από τις αυτοματοποιημένες πληροφορίες “metadata” , περιέχει και το πεδίο payload. Το payload πρακτικά είναι το μήνυμα που έχουμε προγραμματίσει τον αισθητήριο κόμβο να αποστέλλει. Κατά τη διάρκεια προγραμματισμού του κόμβου είχαμε ορίσει τη μεταβλητή **reading** τύπου String , η οποία εμπεριείχε όλα τα δεδομένα των αισθητήρων , με τη σειρά **« reading = θερμοκρασία αέρα + υγρασία αέρα + ατμοσφαιρική πίεση + θερμοκρασία εδάφους + υγρασία εδάφους ».**

Κατά τη λήψη του πρώτου payload μηνύματος διαβάστηκαν από την lcd και το Serial monitor οι παρακάτω τιμές : "θερμοκρασία αέρα": 25.02°C, "υγρασία αέρα": 45.18%, "ατμοσφαιρική πίεση": 934.19HPa, " θερμοκρασία εδάφους ": 24.5°C " υγρασία εδάφους ": 0% . Επομένως μέσω της συγχώνευσης των δεδομένων που πραγματοποιήθηκε μέσω της συνάρτησης getValues() , η μεταβλητή

reading πήρε την τιμή **’ reading = 25.0245.18934.1924.50‘** → reading = 25.02+45.18+934.19+24.5+0‘. Παρόλα αυτά το payload μήνυμα που έφτασε στο TTN δεν δείχνει κατανοητό καθώς είναι της μορφής «32 35 2E 30 32 34 35 2E 31 38 39 33 34 2E 31 39 32 34 2E 35 30 30 2E 30 30».

Εικόνα 43: Πρώτο uplink μήνυμα

Στην πραγματικότητα παρατηρείται πως πρόκειται για ένα κωδικοποιημένο μήνυμα συνολικού μήκους 25 bytes. Το κάθε byte περιέχει τις μετρήσεις των αισθητήρων στη μορφή String που εστάλη από τον κόμβο χωρίς όμως κάποια επεξεργασία.



Εικόνα 44: Μετατροπή uplink μηνύματος

2.6.1 Αποκωδικοποίηση μηνύματος (TTN decoder)

Την αποκωδικοποίηση του μηνύματος αναλαμβάνει η συνάρτηση του **decoder** η οποία εκτελείται στο The Things Network [32] και είναι γραμμένη σε JavaScript. Η συνάρτηση δέχεται το payload [33] μήνυμα (uplink message) που εστάλη από τον αισθητήριο κόμβο σε μορφή πίνακα byte και επιστρέφει ένα Json Object με τις αποκωδικοποιημένες τιμές. Ακολουθεί η συνάρτηση του decoder που αναπτύχθηκε για τις ανάγκες της συγκεκριμένης διπλωματικής εργασίας σε JavaScript.

```
function Decoder(bytes, port) {

var result1 = 0 ; //thermokrasia bme
```

```
var result2 = 0 ; //ygrasia bme
var result3 = 0; //presure bme
var result4 = 0; //thermokrasia ds
var result5 = 0; //ground moisture

for (var i1 = 0; i1 < 5; i1++) {
  result1 += String.fromCharCode(parseInt(bytes[i1]));  }

for (var i2 = 5; i2 < 10; i2++) {
  result2 += String.fromCharCode(parseInt(bytes[i2]));  }

for (var i3 = 10; i3 < 16; i3++) {
  result3 += String.fromCharCode(parseInt(bytes[i]));   }

//gia ground
for (var i4 = 16; i4 < 21; i4++) {
  result4 += String.fromCharCode(parseInt(bytes[i4]));  }

for (var i5 = 21; i5 < bytes.length; i5++) {
  result5 += String.fromCharCode(parseInt(bytes[i5]));  }

result1 = parseFloat(result1);
result2 = parseFloat(result2);
result3 = parseFloat(result3);
result4 = parseFloat(result4);
result5 = parseFloat(result5);

return
{ Air_Temperature: result1 ,
  Air_Humidity: result2 ,
  Air_Pressure: result3 ,
  Ground_Temperature: result4 ,
  Ground_Moisture: result5 ,     };         }
```

- Η συνάρτηση Decoder δέχεται δυο παραμέτρους , το μήνυμα **bytes** και την τιμή **port** (1 έως 223).
- Έπειτα δημιουργεί και αρχικοποιεί τις μεταβλητές result1-5 οι οποίες χειρίζονται στη συνέχεια τα διαφορετικά δεδομένα των αισθητήρων.
- Στη συνέχεια διαχωρίζει το μήνυμα σύμφωνα με την δομή του Payload μηνύματος , δηλαδή, από το πρώτο byte έως το πέμπτο είναι γνωστό πως υπάρχει η πληροφορία της ατμοσφαιρικής θερμοκρασίας , επομένως στη μεταβλητή result1 ένα αποθηκεύται η συγκεκριμένη πληροφορία. Αντίστοιχα την μεταβλητή result2 αποθηκεύεται η πληροφορία της ατμοσφαιρικής υγρασίας (έκτο έως δέκατο byte), στην result3 αποθηκεύεται η πληροφορία της ατμοσφαιρικής πίεσης (ενδέκατο έως δέκατο έκτο byte) , στην result4 αποθηκεύεται η πληροφορία της θερμοκρασίας εδάφους (δέκατο έβδομο έως εικοστό πρώτο) και στην result5 αποθηκεύεται η πληροφορία της υγρασίας εδάφους (εικοστό δεύτερο έως το τελευταίο byte).

- Οι τιμές των result1-5 μετατρέπονται σε τύπου float μέσω της parseFloat() .
- Τέλος επιστρέφονται οι τιμές των αισθητήρων μέσω των πεδίων Air_Temperature, Air_Humidity, Air_Pressure, Ground_Temperature και Ground_Moisture.

Μετά την υλοποίηση της παραπάνω συνάρτησης αποκωδικοποίησης το The Things Network είναι σε θέση να προβάλει σε σωστή μορφή όλα τα δεδομένα που προέρχονται από τον αισθητήριο κόμβο. Πρέπει να επισημανθεί ότι τα δεδομένα δεν αποθηκεύονται στο TTN καθώς δεν αποτελεί βάση δεδομένων , το The Things Network είναι υπεύθυνο μόνο για τη δρομολόγησή τους σε συσκευές και εφαρμογές. Σε επόμενο κεφάλαιο, θα ακολουθήσει αναλυτική εξήγηση της προώθησης των δεδομένων μέσω MQTT στη βάση δεδομένων και τις υπηρεσίες που αναπτύχθηκαν για τις ανάγκες της διπλωματικής εργασίας.

Εικόνα 45: Τελική μορφή μηνύματος

Κεφάλαιο 3ο - Αξιοποίηση δεδομένων του αισθητήριου κόμβου LoRa

3.1 Εγκατάσταση InfluxDB και Plugins (Windows)

Οι αρχικές δοκιμές της καταγραφής των δεδομένων στην InfluxDB με προώθηση μέσω του Telegraf , έγιναν σε περιβάλλον Windows 10, επίσης έγινε εγκατάσταση του plugin Chrograf για την άμεση οπτικοποίηση της InfluxDB.

3.1.1 Εγκατάσταση InfluxDB

Για την εγκατάσταση της InfluxDB ακολουθήθηκαν με τη σειρά τα παρακάτω βήματα:

1. Εγκατάσταση της πιο πρόσφατης έκδοσης μέσω της επίσημης ιστοσελίδας https://portal.influxdata.com/downloads/ .
2. Αποσυμπίεση του αρχείου στη θέση C:\Program Files\InfluxData.
3. Δημιουργία φακέλου **influxdb-data** και υποφακέλων **meta, data** και **wal** στον κατάλογο C:\Program Files\InfluxData.
4. Τροποποίηση του αρχείου **influxdb.conf** (C:\Program Files\InfluxData\influxdb-1.8.3-1).
    i. [meta]
       # Where the metadata/raft database is stored
       dir = "C:\\Program Files\\InfluxData\\influxdb-data\\meta"
    ii. [data]
       # The directory where the TSM storage engine stores TSM files.
       dir = "C:\\Program Files\\InfluxData\\influxdb-data\\data"
    iii. # The directory where the TSM storage engine stores WAL files.
       wal-dir = "C:\\Program Files\\InfluxData\\influxdb-data\\wal"
    iv. [http]
       # Determines whether HTTP endpoint is enabled.
       enabled = true
       # The bind address used by the HTTP service.
       bind-address = ":8086"
       # Determines whether HTTP request logging is enabled.
       log-enabled = true

Η InfluxDB εκκινείται μέσω της εντολής **influxd** στο CMD (εκτέλεση σαν διαχειριστής), στον κατάλογο “CD C:\Program Files\InfluxData\influxdb-1.8.3-1”

3.1.2 Εγκατάσταση Chronograf

Για την εγκατάσταση του Chronograf ακολουθήθηκαν με τη σειρά τα παρακάτω βήματα:

1. Εγκατάσταση της πιο πρόσφατης έκδοσης μέσω της επίσημης ιστοσελίδας https://portal.influxdata.com/downloads/ .
2. Αποσυμπίεση του αρχείου στη θέση C:\Program Files\InfluxData.

Το Chronograf εκκινείται μέσω της εντολής **chronograf** στο CMD (εκτέλεση σαν διαχειριστής), στον κατάλογο "CD C:\Program Files\InfluxData\ chronograf-1.8.8-1" και εφόσον δεν έχει αλλαχτεί το configuration η πρόσβαση γίνεται από τη διεύθυνση. http://localhost:8888/ .

3.1.3 Εγκατάσταση Telegraf

Για την εγκατάσταση του Telegraf ακολουθήθηκαν με τη σειρά τα παρακάτω βήματα:

3. Εγκατάσταση της πιο πρόσφατης έκδοσης μέσω της επίσημης ιστοσελίδας https://portal.influxdata.com/downloads/ .
4. Αποσυμπίεση του αρχείου στη θέση C:\Program Files\InfluxData.
5. Ρύθμιση του αρχείου **telegraf.conf** για προώθηση των δεδομένων του TTN μέσω MQTT στην InfluxDB(C:\Program Files\InfluxData\Telegraf) προσθέτοντας το παρακάτω κομμάτι.

```
[[inputs.lora_node]]
servers = ["tcp://eu.thethings.network:1883"]
qos = 0
connection_timeout = "30s"
topics = [ "+/devices/+/up" ]
client_id = "ttn"
username = "seven_node_1"
password = "ttn-account-v2 credentials"
data_format = "json"
```

- Δημιουργία βάσης `Telegraf\lora_node` , για την αποθήκευση των δεδομένων.
- Subscribe στον sever του TTN (MQTT broker) που φιλοξενεί το application στην προκαθορισμένη θύρα.
- Καθορισμός παραμέτρων σύνδεσης.
- Καθορισμός TOPIC
- Προσθήκη διαπιστευτηρίων , username = "Application ID" και password = "default key-ACCESS KEYS" (μέσω του APPLICATION OVERVIEW).
- Καθορισμός μορφής δεδομένων.

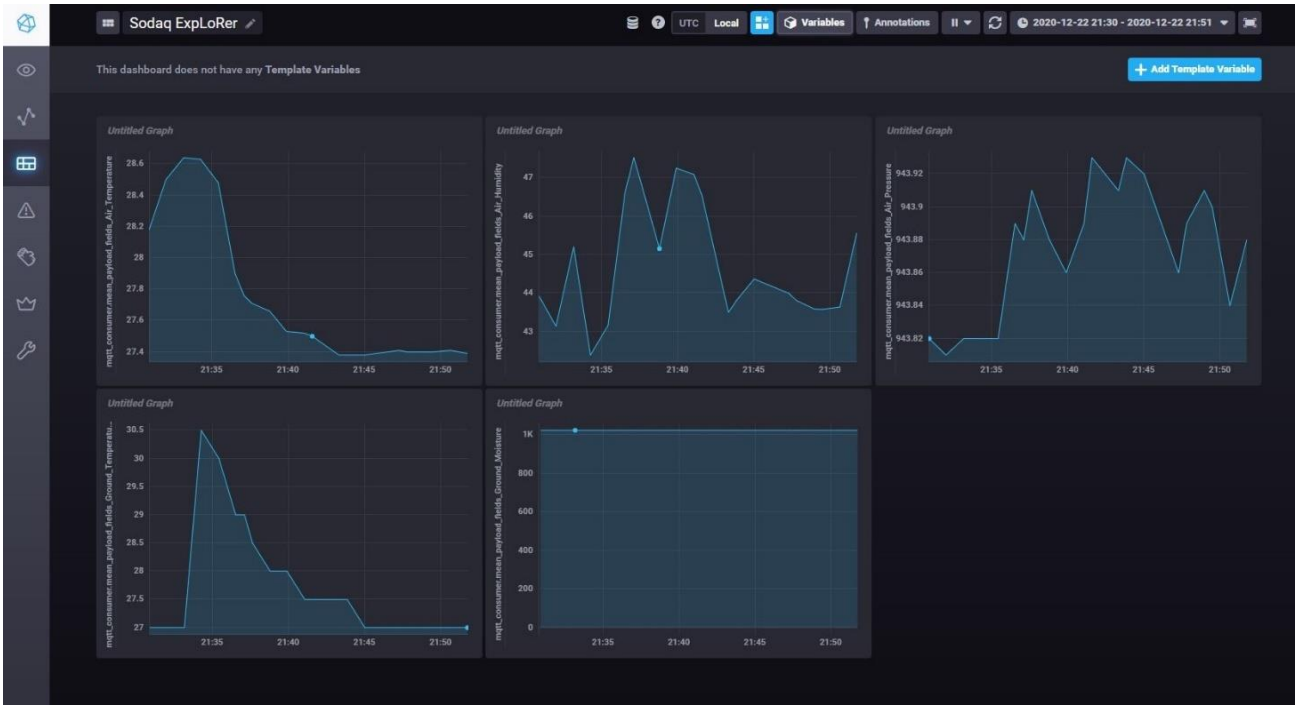
Το Telegraf υποστηρίζει τις παρακάτω εντολές στο CMD (εκτέλεση σαν διαχειριστής), στον κατάλογο "CD C:\Program Files\InfluxData\Telegraf"

Εντολή	Λειτουργία
telegraf.exe --config telegraf.conf –test	Δοκιμή υπηρεσίας
telegraf.exe --service start	Εκκίνηση υπηρεσίας
telegraf.exe --service stop	Τερματισμός υπηρεσίας
telegraf.exe --service install	Εγκατάσταση υπηρεσίας
telegraf.exe --service uninstall	Απεγκατάσταση υπηρεσίας

Πίνακας 14: Linux εντολές Telegraf

3.1.4 Εκτέλεση υπηρεσιών

Μετά από εκτέλεση της influxdb και των plugins τα δεδομένα δείχνουν να φτάνουν κανονικά στη βάση δεδομένων. Η διαδρομή που ακολουθούν τα δεδομένα μέχρι στιγμής είναι, **Αισθητήρες(sodaq)** $\xrightarrow{\text{LoRa}}$ **gateway** $\xrightarrow{\text{Internet}}$ **The Tings Network** $\xrightarrow{\text{Telegraf}}$ **InfluxDB.** Πιο συγκεκριμένα οι αισθητήρες BME280 , DS18b20 και YL-39 YL-69 καταγράφουν τις μετρήσεις μέσω του αισθητήριου κόμβου (Sodaq ExpLoRer) και αποστέλλουν ένα μήνυμα τύπου String που περιέχει συγχωνευμένα όλα τα δεδομένα μέσω LoRa στη συχνότητα των 868MHz. Εφόσον ο κόμβος είναι εντός κάλυψης του δικτύου LoRa , το μήνυμα φτάνει στο gateway, το οποίο με τη σειρά του το προωθεί στο application του The Things Network, όπου ο decoder μετατρέπει το μήνυμα σε ένα json object. Έπειτα το telegraf κάνει εγγραφή μέσω MQTT στη διεύθυνση του server που χειρίζεται το application ώστε να λαμβάνει όλα τα μηνύματα που καταφθάνουν και τα προωθεί στην InfluxDB. Η βάση δεδομένων δημιουργείται στη θέση **telegraf.autogen → LoRa_Node** και τα δεδομένα των μετρήσεων μαζί με τα metadata του κάθε μηνύματος ανανεώνονται αυτόματα στα αντίστοιχα πεδία. Από την εκτέλεση του Chronograf είναι δυνατή η άμεση οπτικοποίηση των μετρήσεων όπως φαίνεται παρακάτω.



Εικόνα 46: Οπτικοποίηση μέσω Chronograf

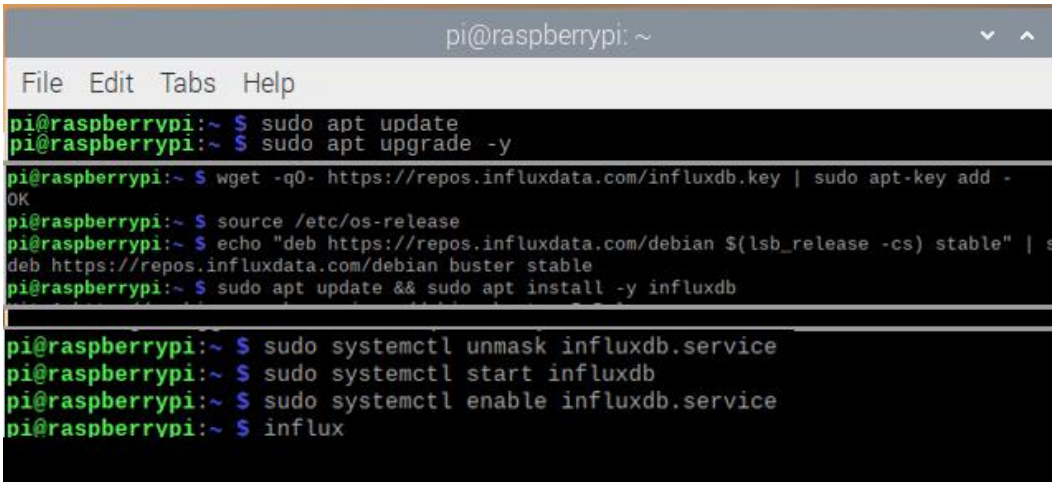
3.2 Δημιουργία εξυπηρετητή (micro-server)

Αν και είναι εφικτό η βάση και τα plugins να εκτελούνται σαν υπηρεσίες σε ένα μηχάνημα windows, για τις ανάγκες της συγκεκριμένης διπλωματικής εργασίας αποφασίστηκε η εγκατάσταση ενός εξυπηρετητή αποκλειστικά για τις απαιτήσεις του αισθητήριου κόμβου και των λειτουργιών της εφαρμογής. Τον εξυπηρετητή αποτελεί μια συσκευή Raspberry Pi 4 καθώς είναι εξαιρετική λύση με μικρό κόστος.

Το Raspberry Pi 4 που χρησιμοποιήθηκε είναι μια υπολογιστική συσκευή πολύ μικρού μεγέθους, διαθέτει επεξεργαστή 1.5GHz quad-core αρχιτεκτονικής ARM Cortex-A72 CPU στα 64-bit, 4gb μνήμης ram καθώς και διασύνδεση στο διαδίκτυο μέσω WiFi και gigabit Ethernet. Το λειτουργικό σύστημα που εκτελείται το Raspberry Pi είναι το Raspberry Pi OS (Raspbian) , πρόκειται για ένα λειτουργικό σύστημα βασισμένο σε Debian και υποστηρίζει τις αντίστοιχες Linux εντολές μέσω του terminal.

Αρχικά έγινε εγκατάσταση της InfluxDB, του Telegraf , καθώς και του Plugin Grafana. Η εγκατάσταση του Chrograf δεν κρίθηκε απαραίτητη μια και τη διεργασία της οπτικοποίησης την ανέλαβε το λογισμικό Grafana.

3.2.1 Εγκατάσταση InfluxDB



Εικόνα 47: Εγκατάσταση InfluxDB

Η εγκατάσταση της InfluxDB έγινε μέσω του Raspberry terminal, παρακάτω εξηγείται με τη σειρά η διαδικασία που ακολουθήθηκε.

- Μέσω των εντολών `sudo apt update` και `sudo apt upgrade` έγινε ενημέρωση και εγκατάσταση των πιο πρόσφατων πακέτων στη συσκευή Raspberry Pi 4.
- Προστέθηκαν τα πιο πρόσφατα repositories της InfluxDB και έπειτα ακολούθησε η ενημέρωση και η εγκατάστασή τους.

```
wget -q -0- https://repos.influxdata.com/influxdb.key | sudo apt-key add -

source /etc/os-release

echo "deb https://repos.influxdata.com/debian $(lsb_release -cs)  stable"
|sudo tee /etc/apt/sources.list.d/influxdb.list

sudo apt update

sudo apt install -y influxdb
```

Η βάση δεδομένων αναμένεται να μπορεί να συλλέγει πάντα δεδομένα μέσω του Telegraf , γι' αυτό είναι σημαντικό να εκτελείται αυτόματα σαν υπηρεσία στον εξυπηρετητή , δηλαδή να ξεκινά η εγγραφή των δεδομένων χωρίς κάποια ενέργεια στην περίπτωση μιας επανεκκίνησης.

- Μέσω των εντολών `sudo systemctl unmask influxdb.service`, `sudo systemctl start influxdb` και `sudo systemctl enable influxdb.service` έγινε εκκίνηση της υπηρεσίας influxdb και της επιτράπηκε η εκτέλεση κατά τη διαδικασία εκκίνησης (boot).

3.2.2 Εγκατάσταση Grafana

Η εγκατάσταση του λογισμικού Grafana έγινε μέσω του Raspberry terminal, παρακάτω εξηγείται με τη σειρά η διαδικασία που ακολουθήθηκε.

- Προστέθηκαν τα πιο πρόσφατα repositories του Grafana και έπειτα ακολούθησε η ενημέρωση και η εγκατάστασή τους.

```
wget -q -0 - https://packages.grafana.com/gpg.key| sudo apt-key add -

echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee
/etc/apt/sources.list.d/grafana.list

sudo apt update

sudo apt install -y grafana
```

Έπειτα έγινε εκκίνηση της υπηρεσίας grafana-server και της επιτράπηκε η εκτέλεση κατά τη διαδικασία εκκίνησης.

```
sudo systemctl unmask grafana-server.service

sudo systemctl start grafana-server

sudo systemctl enable grafana-server.service
```

3.2.3 Εγκατάσταση Telegraf

Η εγκατάσταση του Telegraf έγινε επίσης μέσω του Raspberry terminal, με την παρακάτω διαδικασία.

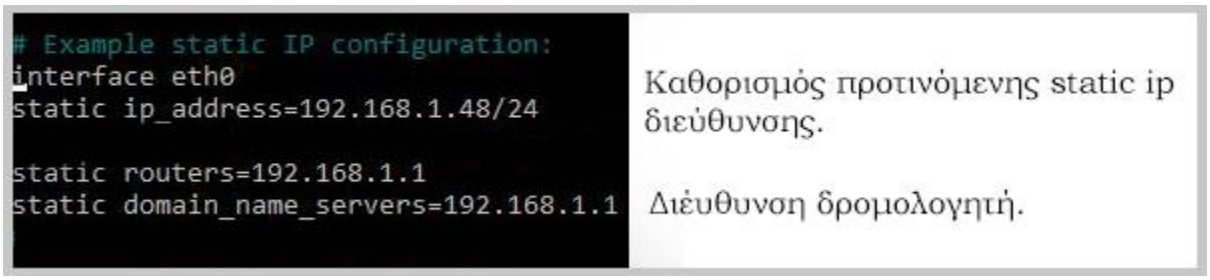
```
wget https://dl.influxdata.com/telegraf/releases/telegraf_1.14.2-
1_armhf.deb

sudo dpkg -i telegraf_1.14.2-1_armhf.deb

sudo apt update

sudo apt install -y telegraf
```

Ακολούθησε εκκίνηση της υπηρεσίας telegraf και της επιτράπηκε η εκτέλεση κατά τη διαδικασία εκκίνησης.

```
sudo systemctl unmask telegraf.service

sudo systemctl start telegraf-server

sudo systemctl enable telegraf-server.service
```

3.3 Ρύθμιση εξυπηρετητή

Το Raspberry Pi από προεπιλογή επιχειρεί να ρυθμίσει την IP διεύθυνσή του αυτόματα (dynamic ip), είναι πολύ πιθανό μια τρέχουσα ip διεύθυνση να μην συμπίπτει με την μελλοντική. Στην περίπτωση της συγκεκριμένης διπλωματικής εργασίας, το Raspberry Pi λειτουργεί σαν εξυπηρετητής και είναι προτιμότερο να διαθέτει στατική IP διεύθυνση για ευκολότερη πρόσβαση στις υπηρεσίες του και απευθείας έλεγχο μέσω του πρωτοκόλλου SSH. Για τους παραπάνω λόγους, η IP του εξυπηρετητή ρυθμίστηκε σε STATIC από το αρχείο `dhcpd.conf` , μέσω του επεξεργαστή κειμένου nano με την εντολή `nano /etc/dhcpcd.conf` .




```
# Example static IP configuration:
interface eth0
static ip_address=192.168.1.48/24

static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Εικόνα 48: Διαμόρφωση αρχείου dhcpcd

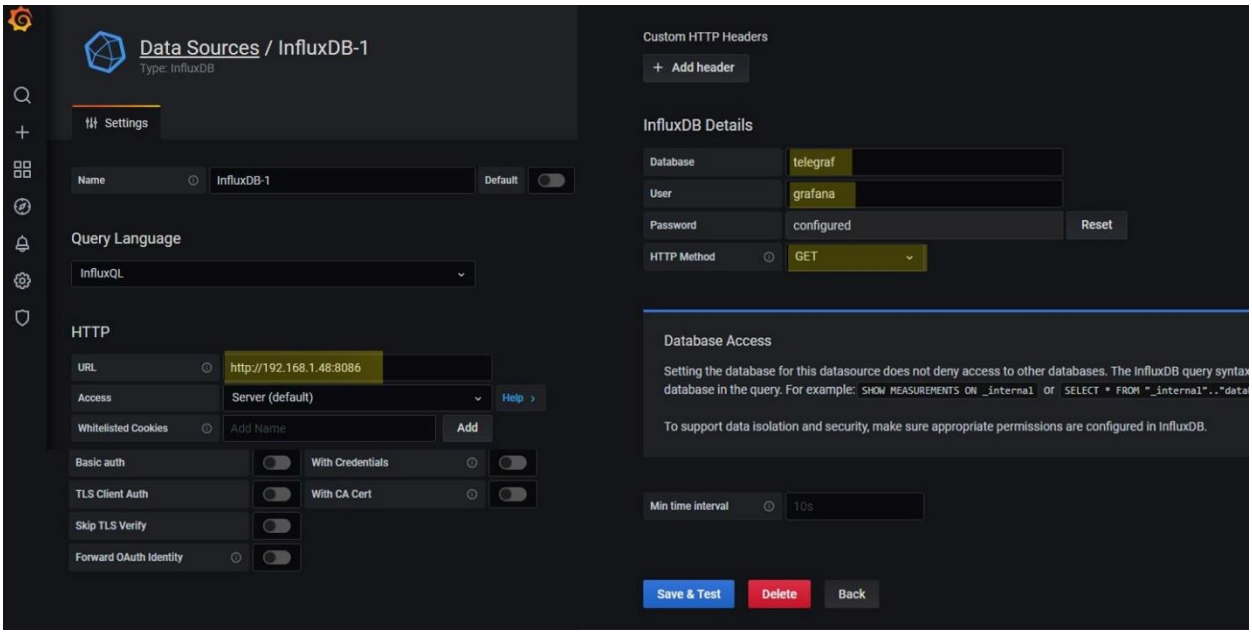
Όπως και στην περίπτωση χρήσης της υπηρεσίας Telegraf σε Windows , έγινε ενημέρωση του αρχείου **telegraf.conf** για την προώθηση των δεδομένων του TTN στην influxdb, προσθέτοντας το ίδιο κομμάτι στο πεδίο των INPUT PLUGINS.

```
[[inputs.mqtt_consumer]]
  servers = ["tcp://eu.thethings.network:1883"]
  qos = 0
  connection_timeout = "30s"
  topics = [ "+/devices/+/up" ]
  client_id = "ttn"
  username = "seven node 1"
  password =
  data_format = "json"
```

Εικόνα 49: Αρχείο telegraf.conf

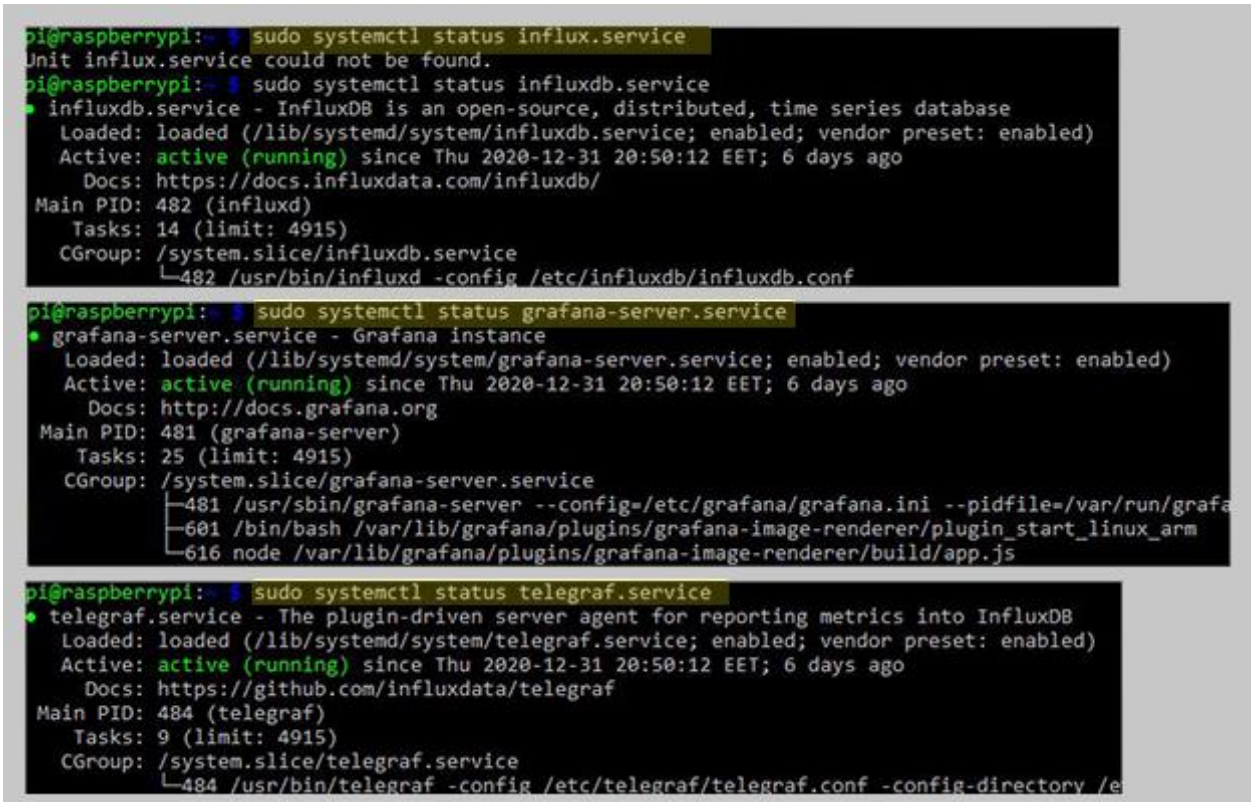
Έπειτα πραγματοποιήθηκε ρύθμιση του Grafana. Η προκαθορισμένη θύρα εκτέλεσής του είναι η `3000` στην διεύθυνση του εξυπηρετητή , δηλαδή στην στατική ip που ορίστηκε προηγουμένως (`http://192.168.1.48:3000`). Προστέθηκε η βάση της influxdb σαν πηγή δεδομένων στην οποία προωθούνται τα δεδομένα μέσω του Telegraf.

- Ορίστηκε η URL της πηγής , δηλαδή της διεύθυνσης που εκτελείτε η InfluxDB.
- Προσδιορίστηκε η συγκεκριμένη βάση και ο χρήστης της.
- Καθορίστηκε η μέθοδος HTTP.



Εικόνα 50: Διαμόρφωση πηγής δεδομένων

Ακολούθησε επανεκκίνηση όλων των υπηρεσιών, InfluxDB , Grafana και Telegraf μέσω της εντολής `sudo systemctl reload` **`<όνομα_υπηρεσίας>`**`.service` και πραγματοποιήθηκε έλεγχος της τρέχουσας κατάστασής τους.

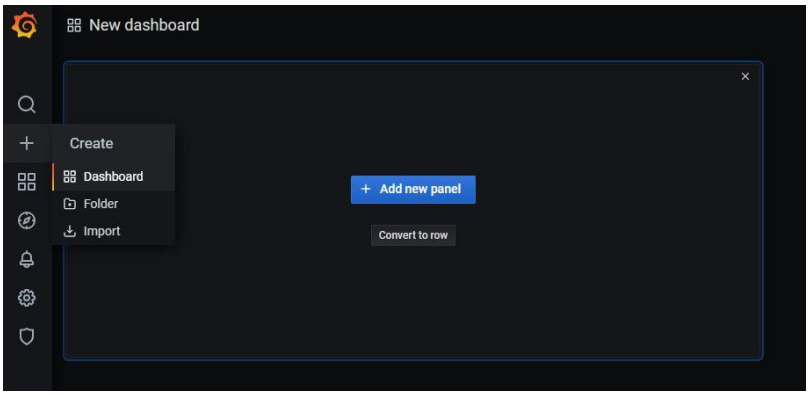


Εικόνα 51: Δραστηριότητα υπηρεσιών

3.4 Οπτικοποίηση μέσω της Grafana

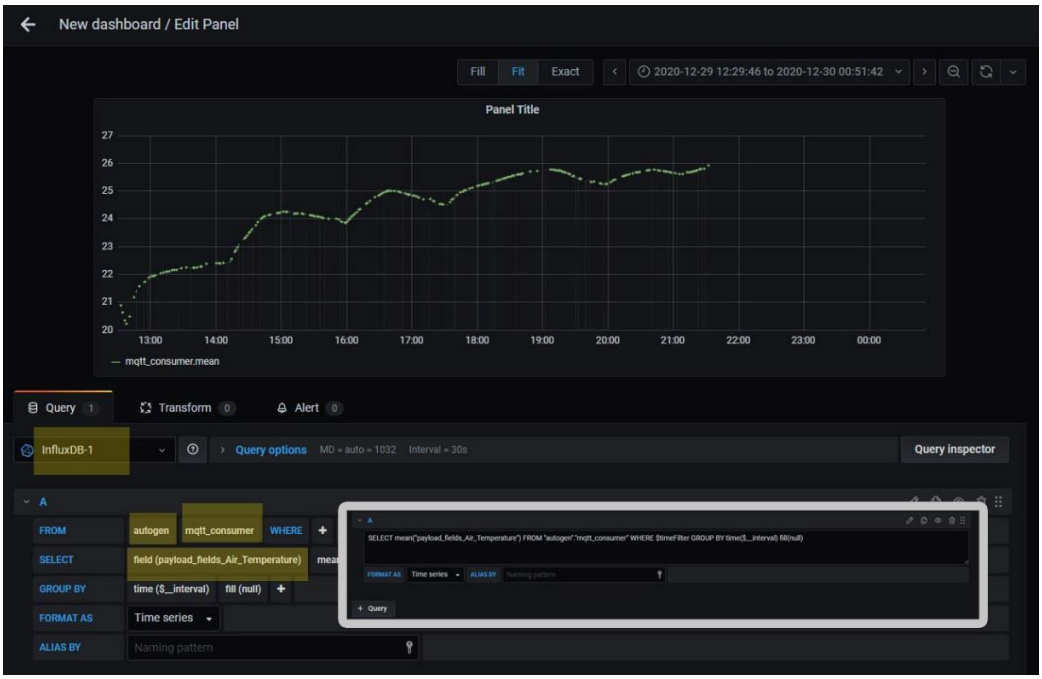
Μετά την επιτυχημένη ρύθμιση του εξυπηρετητή και τον έλεγχο λειτουργίας όλων των υπηρεσιών ακολούθησε η διαδικασία της οπτικοποίησης μέσω του Grafana Plugin. Παρακάτω η διαδικασία δημιουργίας ενός Grafana Dashboard:

- Είσοδος στο Grafana μέσω της διεύθυνσης `http://192.168.1.48:3000`.
- Δημιουργία νέου Dashboard και νέου panel.



Εικόνα 52: Δημιουργία πίνακα ελέγχου

- Data querying μέσω της user friendly δυνατότητας με επιλογή των κατάλληλων πεδίων και του χρονικού εύρους μετρήσεων. Στο συγκεκριμένο panel έγινε οπτικοποίηση της ατμοσφαιρικής θερμοκρασίας , με την επιλογή της πηγής δεδομένων που ορίστηκε κατά την ρύθμιση του εξυπηρετητή. Το κατάλληλο query δημιουργείται αυτόματα και μπορεί να τροποποιηθεί.



Εικόνα 53: Data querying μέσω Grafana

Η παραπάνω διαδικασία επαναλήφθηκε για όλα τα πεδία του αισθητήριου κόμβου καθώς επίσης υλοποιήθηκαν και γραφήματα (panel) για την οπτικοποίηση της τελευταίας μέτρησης του κάθε αισθητήρα , με σκοπό τη δημιουργία ενός ενοποιημένου Dashboard για όλες τους αισθητήρες, με ονομασία LoRa Node_daily που αποτυπώνει τις μετρήσεις του τελευταίου εικοσιτετράωρου.

Εικόνα 54: Ολοκληρωμένος πίνακας ελέγχου Grafana

Αφού αποθηκευτούν οι ρυθμίσεις που πραγματοποιήθηκαν, το συγκεκριμένο Grafana Dashboard είναι πλέον άμεσα προσβάσιμο από όλες τις συσκευές μέσω της διεύθυνσης του εξυπηρετητή στο τοπικό δίκτυο με τη χρήση των διαπιστευτηρίων εισόδου.

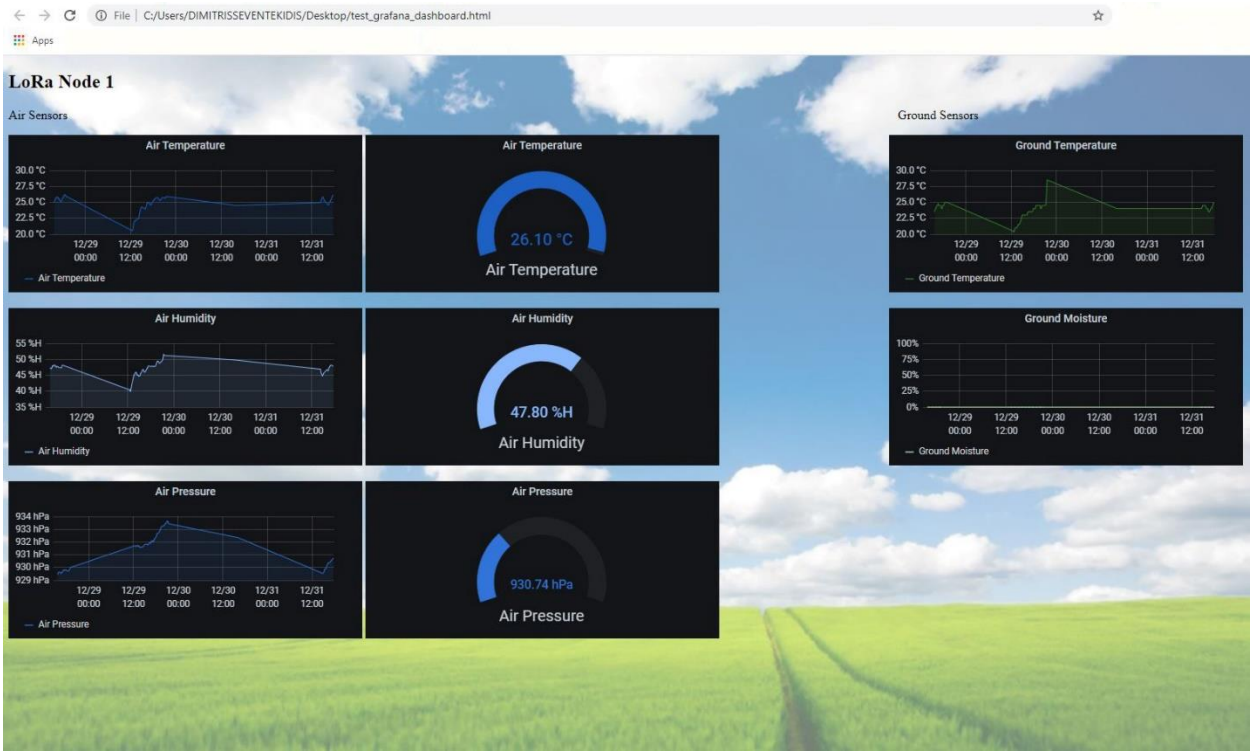
Είναι χρήσιμο ένα Grafana Dashboard ή συγκεκριμένα Panel αυτού να μπορούν να ενσωματωθούν, όπως για παράδειγμα σε ιστοσελίδες ή τρίτες εφαρμογές. Η Grafana υποστηρίζει αυτή τη δυνατότητα μετά από κατάλληλη ρύθμιση. Για να επιτραπεί η ενσωμάτωση απαιτείται η τροποποίηση του αρχείου `grafana.ini` στη θέση `/etc/grafana/`.

- Ενεργοποίηση του επεξεργαστή κειμένου nano , `nano /etc/grafana/grafana.ini` .

- Επεξεργασία του αρχείου , θέτοντας το `allow embedding` σε `True` και αφαιρώντας τους χαρακτήρες <;> :

```
allow_embedding = true

auth.anonymous
enabled = true
org_name = <<org name>>
org_role = Viewer
```

Έπειτα από την τροποποίηση του αρχείου `grafana.ini` και επανεκκίνηση των υπηρεσιών ακολούθησε γρήγορη δοκιμή υλοποιώντας μια απλή ιστοσελίδα σε html και παρατηρήθηκε πως τα γραφήματα ενσωματώνονται με επιτυχία.



Εικόνα 55: Δοκιμή ενσωμάτωσης γραφημάτων σε HTML

3.5 Εφεδρική αποθήκευση (backup) μέσω influxdb cloud

Για τις ανάγκες αποθήκευσης των δεδομένων του αισθητήριου κόμβου, αποφασίστηκε εκτός από την βάση δεδομένων που εγκαταστάθηκε στον micro-server να χρησιμοποιηθεί και εφεδρική μέσω cloud. Πιο συγκεκριμένα χρησιμοποιήθηκε η υποδομή που προσφέρει η ίδια η InfluxData μέσω του **cloud2.influxdata.com**. Την προώθηση των δεδομένων στο cloud ανέλαβε και πάλι το telegraf όπως θα εξηγηθεί παρακάτω.

- Αφού πραγματοποιήθηκε εγγραφή στην πλατφόρμα, δημιουργήθηκε κατάλληλο bucket και δόθηκε μοναδικό token για χρήση από το Telegraf.
- Ακολούθησε τροποποίηση του Telegraf, προσθέτοντας την διεργασίας προώθησης στο cloud. Συγκεκριμένα, δόθηκαν τα διαπιστευτήρια και το mail της εγγραφής καθώς και η διεύθυνση προορισμού και το συγκεκριμένο bucket.

```
[[outputs.influxdb_v2]]
  urls = ["https://us-central1-1.gcp.cloud2.influxdata.com"]

  ## Token for authentication.
  token =

  organization = "ece01071@uowm.gr"
  bucket = "LoRaWAN_Metrics"
```

Εικόνα 56: Telegraf output για προώθηση στο InfluxDB Cloud

Μετά την επανεκκίνηση του telegraf τα δεδομένα άρχισαν να αποθηκεύονται και στο cloud, η online πλατφόρμα της InfluxData προσφέρει εκτός από την αποθήκευση δεδομένων χρονοσειρών δυνατότητες οπτικοποίησης , ειδοποίησης και εφαρμογής κατάλληλων ερωτημάτων (queries), επίσης δίνει τη δυνατότητα της άμεσης αποθήκευσης των δεδομένων στον τοπικό υπολογιστή του χρήστη μέσω υπολογιστικών φύλλων Excel.

Εικόνα 57: Πίνακας ελέγχου InfluxDB Cloud

3.6 Αλληλεπίδραση με τον τελικό χρήστη μέσω Telegram

Για την αλληλεπίδραση του τελικού χρήστη με τα δεδομένα μετρήσεων του αισθητήριου κόμβου μέσω του διαδικτύου αποφασίστηκε να χρησιμοποιηθεί η cloud εφαρμογή ανταλλαγής μηνυμάτων Telegram. Πιο συγκεκριμένα μέσω της δυνατότητας των Telegram bots υλοποιήθηκε υπηρεσία σε python η οποία εκτελείται στον ίδιο εξυπηρετητή Raspberry Pi μαζί με τις υπόλοιπες υπηρεσίες. Η συγκεκριμένη υπηρεσία είναι σε θέση να αποστέλλει τα δεδομένα του αισθητήριου κόμβου μέσω της βάσης δεδομένων καθώς και γραφήματα που εξάγει το λογισμικό Grafana.

Το λογισμικό Grafana διαθέτει πρόσθετο-plugin “Rendering” με σκοπό την εξαγωγή γραφημάτων για αποστολή σε τρίτες εφαρμογές σαν αρχείο .png ή .jpg, το plugin όμως προσφέρεται απευθείας από την Grafana Labs για λειτουργικά συστήματα Linux, Windows και Mac OS αλλά όχι για εγκατάσταση σε αρχιτεκτονικές Debian όπως αυτή του Raspberry Pi OS με βάση το οποίο λειτουργεί ο εξυπηρετητής (micro-server) που εξυπηρετεί τις ανάγκες της συγκεκριμένης διπλωματικής εργασίας.

Για τους παραπάνω λόγους η εγκατάσταση του rendering plugin του λογισμικού Grafana στο Raspberry Pi αποτέλεσε πρόκληση στη διάρκεια της διπλωματικής εργασίας, καθώς όμως η ανάγκη του κρίθηκε απαραίτητη, δόθηκε ιδιαίτερη σημασία στην εύρεση λύσης. Τελικά η προσπάθεια κατέληξε στην επιτυχή εγκατάσταση του plugin με τη διαδικασία που περιγράφεται στην υπό ενότητα 3.6.1 .

3.6.1 Εγκατάσταση του Grafana rendering plugin σε λειτουργικό Raspberry Pi OS

Για την εγκατάσταση του Grafana rendering plugin στον εξυπηρετητή (Raspberry Pi 4) ακολουθήθηκε η παρακάτω διαδικασία, εξαιρετικά χρήσιμες αποδείχθηκαν οι συμβουλές της κοινότητας του Openhab.org [34] :

- Εγκατάσταση του Plugin.
```
apt-get update
apt-get install git
cd /var/lib/grafana/plugins
git clone
https://github.com/grafana/grafana-image-
renderer
cd grafana-image-renderer
npm -g install yarn
```

- Ενημέρωση του npm.
```
npm -g install npm
```

- Εγκατάσταση των εργαλείων Yarn και Typescript.
```
npm -g install yarn
npm -g install typescript
```

- Αλλαγή της `unsafe-perm` σε true.
```
npm config set unsafe-perm=true
```

- Εγκατάσταση πρόσθετων.
```
yarn install --pure-lockfile
yarn run build
```

```
cp plugin_start_linux_amd64 plugin_start_linux_arm
```

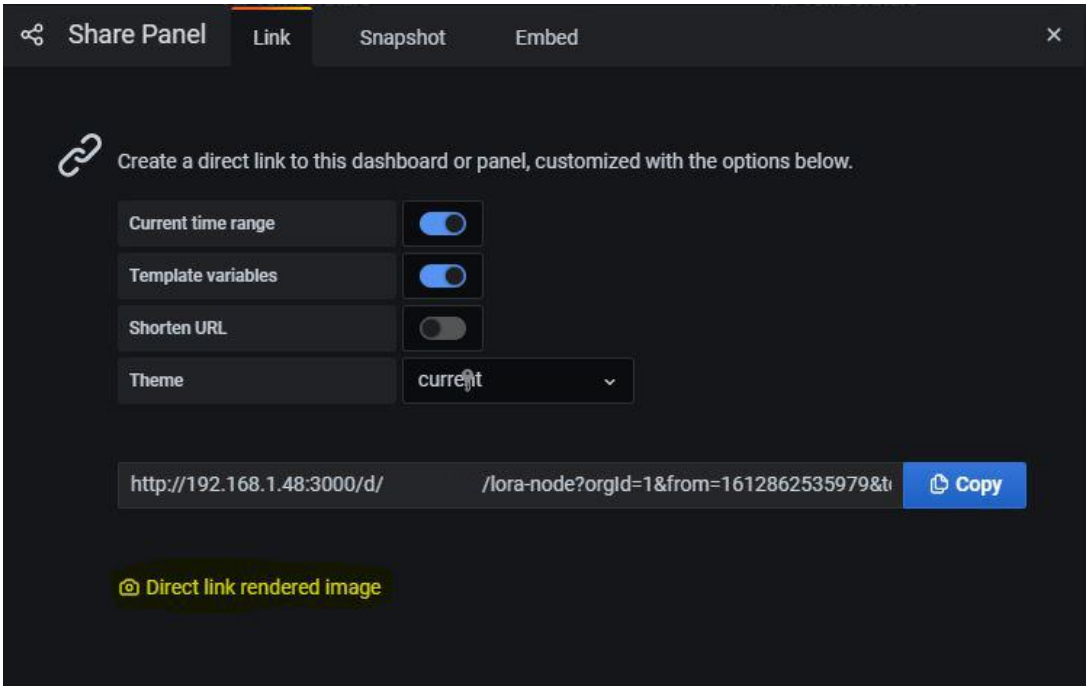
- Εγκατάσταση του Chromium Browser.
```
sudo apt-get install chromium-browser
```

- Ρύθμιση του plugin_start_linux προσθέτοντας την παρακάτω γραμμή.
```
export GF_RENDERER_PLUGIN_CHROME_BIN="/usr/bin/chromium-browser"
```

- Προσθήκη της παρακάτω γραμμής στο αρχείο grafana.ini, στη θέση /etc/grafana/grafana.ini.
```
[plugins]
allow_loading_unsigned_plugins = "grafana-image-renderer"
```

- Τέλος ακολουθεί επανεκκίνηση της υπηρεσίας Telegraf και μετά από λίγο έλεγχος για την επιτυχή εκτέλεσή της.
```
sudo systemctl restart grafana-server.service
sudo systemctl status grafana-server.service
```

Στη θέση /var/log/grafana το plugin δημιουργείται το log file, στο οποίο καταγράφονται όλες οι απόπειρες rendering. Κάνοντας είσοδο στην διεύθυνση που εξυπηρετεί το λογισμικό Grafana παρατηρείται πως πλέον είναι διαθέσιμη η επιλογή του rendering.



Εικόνα 58: Ενεργοποίηση δυνατότητας rendering

Τέλος ακολουθεί δοκιμή με στόχο την εξαγωγή ενός Grafana panel σε μορφή png. Δίνεται η παρακάτω εντολή με βάση την οποία θα πρέπει να εκτελεστεί το rendering plugin και να εξάγει το **Panel 4** από το Dashboard **lora-node** στο αρχείο "panel4.png":

curl "http://192.168.1.48:3000/**render**/d/loraNode/lora-node?orgId=1&from=1609179425224&to=1609352225224&viewPanel=4" > panel4.png

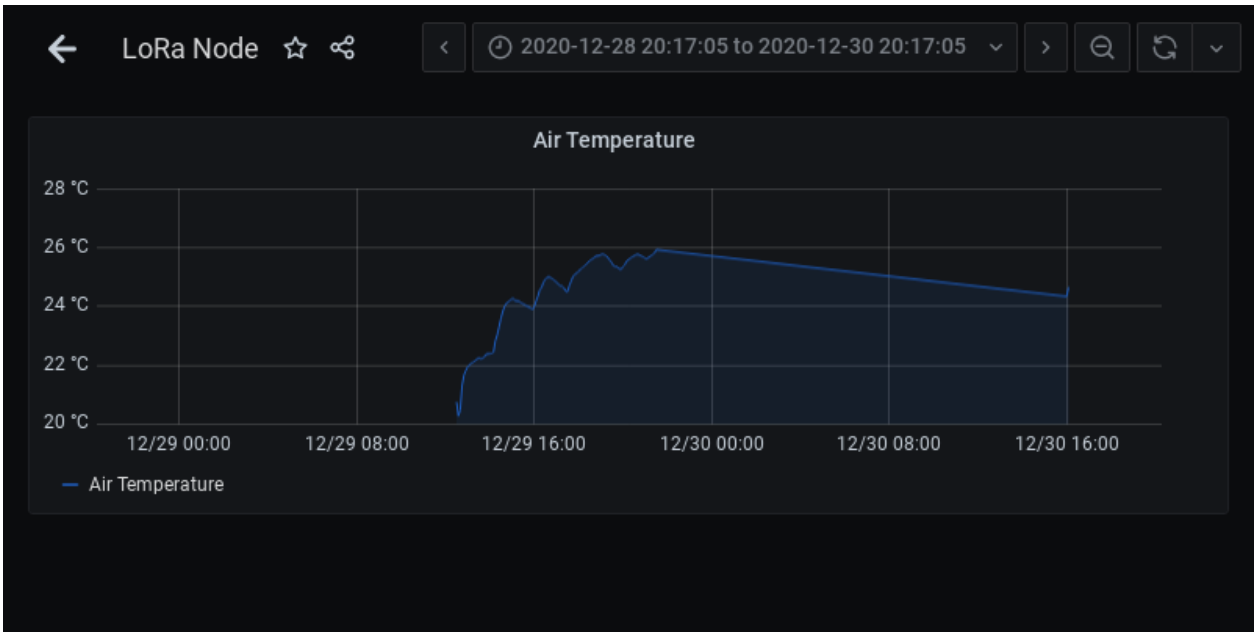
Η εντολή εκτελείται με επιτυχία με χρόνο εκτέλεσης 6 δευτερόλεπτα, και δημιουργείται στο αρχικό directory το αρχείο panel4.png το οποίο αποτυπώνει το αντίστοιχο πάνελ του Grafana την τρέχουσα στιγμή.

```
pi@raspberrypi:~ $ curl [redacted] > panel4.png
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 23015  100 23015    0     0   3609      0  0:00:06  0:00:06 --:--:--  5515
```

Εικόνα 59: Χρόνος εκτέλεσης rendering

```
pi@raspberrypi: /var/log/grafana
  GNU nano 3.2

t=2021-02-13T15:49:47+0200 lvl=info msg=Rendering
t=2021-02-13T15:54:32+0200 lvl=info msg=Rendering        &viewPanel=4"
t=2021-02-13T16:04:14+0200 lvl=info msg=Rendering        &viewPanel=4"
t=2021-02-13T16:05:57+0200 lvl=info msg=Rendering        &viewPanel=4"
```

Εικόνα 60: Αρχείο log file Grafana



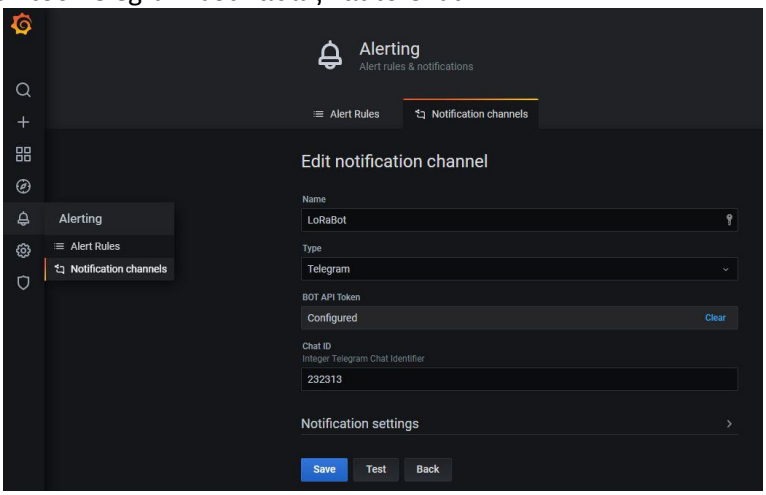
Εικόνα 61: Παράδειγμα γραφήματος rendering

3.6.2 Ρύθμιση ειδοποιήσεων μέσω του Grafana

Αφού προσδιορίστηκαν τα όρια των τιμών που δεν πρέπει να ξεπεραστούν, ρυθμίστηκαν οι ειδοποιήσεις που θα αποστέλλει το Grafana στον τελικό χρήστη. Αξίζει να σημειωθεί πως είναι πολύ σημαντικό να γίνεται σωστός προσδιορισμός των ορίων μια τιμής και πότε θα πρέπει να ειδοποιείται ο τελικός χρήστης, καθώς δεν πρέπει να συσσωρεύονται μη αναγκαίες ειδοποιήσεις που δεν απαιτούν την παρέμβαση του ανθρώπινου παράγοντα.

Το Grafana έχει την δυνατότητα να αποστέλλει ειδοποιήσεις με διάφορους τρόπους, συμπεριλαμβανομένου του Telegram , email αλλά και άλλων εφαρμογών. Αποφασίστηκε να χρησιμοποιηθεί η δυνατότητα του Telegram όπου ένα Telegram bot θα στέλνει τις κατάλληλες ειδοποιήσεις.

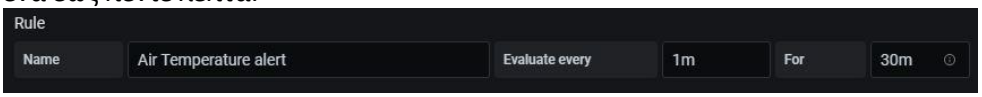
Αρχικά έγινε ρύθμιση ενός καναλιού ειδοποίησης (notification channel), όπου καταχωρήθηκε το API token του Telegram bot καθώς και το Chat ID.



Εικόνα 62: Ρύθμιση καναλιού ειδοποίησης

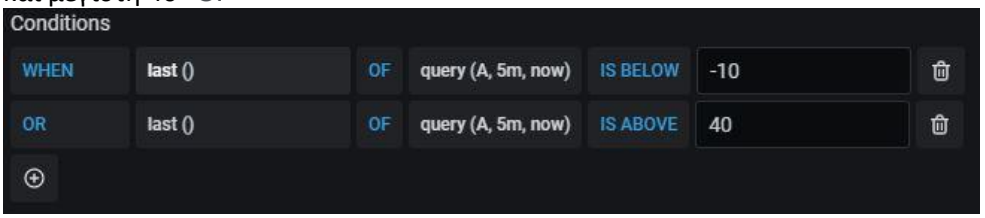
Στη συνέχεια επιλέγοντας το αντίστοιχο πάνελ της κάθε μέτρησης πραγματοποιήθηκαν οι κατάλληλες ρυθμίσεις. Ακολουθεί η ρύθμιση των ειδοποιήσεων της ατμοσφαιρικής θερμοκρασίας μέσω του πεδίου **Alert**.

- Έγινε προσδιορισμός του χρόνου αξιολόγησης των δεδομένων από ένα έως πέντε λεπτά.



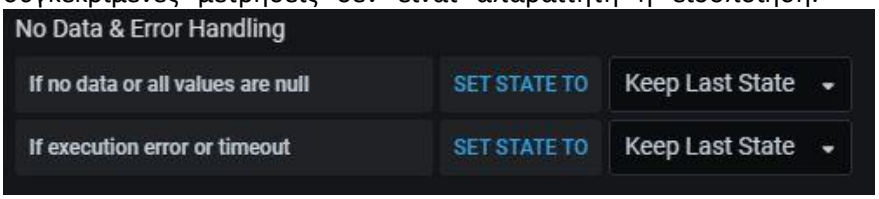
Εικόνα 63: Προσδιορισμός χρόνου

- Έπειτα δόθηκαν τα όρια δόθηκαν τα άνω και κάτω όρια της ατμοσφαιρικής θερμοκρασίας ως εξής, ελάχιστη θερμοκρασία -10 °C και μέγιστη 40 °C.



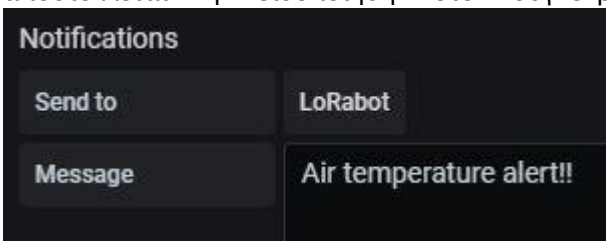
Εικόνα 64: Καθορισμός ορίων

- Αποφασίστηκε στις περιπτώσεις No Data και Error Handling να παραμένει η τελευταία επιβεβαιωμένη τιμή καθώς στις συγκεκριμένες μετρήσεις δεν είναι απαραίτητη η ειδοποίηση.



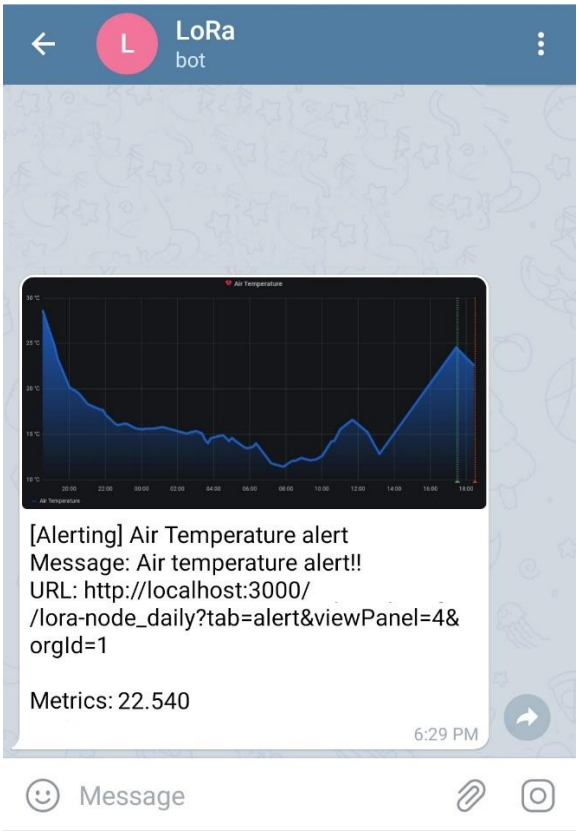
Εικόνα 65: Error Handling

- Τέλος προσδιορίστηκε το μήνυμα και το κανάλι που θα αποστέλλεται η ειδοποίηση στο συγκεκριμένο σενάριο.



Εικόνα 66: Ορισμός μηνύματος

Έπειτα πραγματοποιήθηκε η ίδια διαδικασία για τα πάνελ των υπόλοιπων δεδομένων και ακολουθήσε έλεγχος για την σωστή αποστολή των ειδοποιήσεων στο κανάλι Telegram.



Εικόνα 67: Παράδειγμα ειδοποίησης

3.6.3 Προγραμματισμός υπηρεσίας Lora_server

Σε αυτή την υπό ενότητα θα γίνει αναλυτική περιγραφή προγραμματισμού της υπηρεσίας “Lora_server”, που υλοποιήθηκε σε γλώσσα Python για τις ανάγκες αλληλεπίδρασης με τον τελικό χρήστη μέσω του Telegram στο ίδιο κανάλι που ρυθμίστηκαν και οι ειδοποιήσεις του Grafana.

- Αρχικά έγινε η δημιουργία ενός Telegram bot με όνομα **LoRa_Node** ακολουθώντας τη διαδικασία που έχει περιγραφεί στην υπό ενότητα «Telegram-bot 1.5.4».
- Γίνεται εγκατάσταση της βιβλιοθήκης telepot για Python, στην οποία θα εισαχθεί το χαρακτηριστικό “bot token” και έπειτα ακολουθεί ο προγραμματισμός της υπηρεσίας “Lora_server” σε Python.

Ο κώδικας χωρίζεται σε δύο μέρη, αυτό των δηλώσεων - βοηθητικών συναρτήσεων και αυτό της κύριας συνάρτησης. Η κύρια συνάρτηση εκτελείται συνεχώς και βρίσκεται σε αναμονή για λήψη καινούριου μηνύματος από το χρήστη και ακολουθώντας μια δομή ελέγχου απαντά επιστρέφοντας την ζητούμενη πληροφορία.

3.6.3.1 Δηλώσεις – βοηθητικές συναρτήσεις

```
import time
import random
import datetime
import telepot
import os
import urllib
from telepot.loop import MessageLoop
from gpiozero import CPUTemperature
from influxdb import InfluxDBClient
```

- Γίνεται δήλωση όλων των απαραίτητων μεταβλητών για την εκτέλεση του κώδικα.

```
client = InfluxDBClient(host='192.168.1.48', port='8086')
client.switch_database('telegraf')
```

- Ακολουθεί ο καθορισμός της βάσης δεδομένων από την οποία αντλούνται τα δεδομένα που χειρίζεται το Telegram bot.

```
def convert(seconds):
    days = seconds // (60*60*24)
    seconds = seconds % (24 * 3600)
    hour = seconds // 3600
    seconds %= 3600
    minutes = seconds // 60
    seconds %= 60
    return "\xF0\x9F\x95\x90 %d Days %d Hrs. %02d Min. and %02d Sec." % (days, hour, minutes, seconds)
```

- Τέλος συντάσσεται η βοηθητική συνάρτηση def convert(seconds), η οποία κατά την κλήση της μετατρέπει τα milliseconds σε String της μορφής “ Ημέρες-Ώρες-Λεπτά-Δευτερόλεπτα”.

3.6.3.2 Κυρίως πρόγραμμα

Το κυρίως πρόγραμμα αποτελείται από τη συνάρτηση def handle(msg), η οποία χειρίζεται τα μηνύματα που καταφθάνουν στο κανάλι Telegram και απαντάει ανάλογα με το ζητούμενο. Η δομή της def handle(msg), συνίσταται από δομή ελέγχου **if…elif…else** η οποία ελέγχει το μήνυμα και εκτελεί το αντίστοιχο block κώδικα μέσα από το οποίο αποστέλλεται μήνυμα στο κανάλι Telegram από το bot μέσω της **bot.sendMessage(chat_id, «μήνυμα»)**. Η συνάρτηση καλείται μέσω της εντολής **MessageLoop(bot, handle).run_as_thread()** εφόσον καταφθάσει νέο μήνυμα από τον χρήστη.

```
def handle(msg):
    if command == '/server_info':
        cpu = CPUTemperature()
        temper1 = float(CPUTemperature())
        uptime1 = float(uptime)
        stat1 = os.system('systemctl is-active --quiet influxdb.service')   #epistrefei 0 gia active
    if stat1 == 0:
        influx = '\xE2\x96\xAA Influxdb is active \xE2\x9C\x85'
    else :
        influx = '\xE2\x96\xAA Influxdb is inactive \xE2\x9D\x8C'
    stat2 = os.system('systemctl is-active --quiet telegraf.service')   #epistrefei 0 gia active
    print(stat2)
    if stat2 == 0:
        telegraf = '\xE2\x96\xAA Telegraf is active \xE2\x9C\x85'
    else :
        telegraf = '\xE2\x96\xAA Telegraf is inactive \xE2\x9D\x8C'
    stat3 = os.system('systemctl is-active --quiet grafana-server.service')   #epistrefei 0 gia active
    if stat3 == 0:
        grafana = '\xE2\x96\xAA Grafana is active \xE2\x9C\x85'
    else :
        grafana = '\xE2\x96\xAA Grafana is inactive \xE2\x9D\x8C'
    n = int(float(uptime))
    bot.sendMessage(chat_id, '\xF0\x9F\x92\xBB SERVER is running at: ' + str(temper) + '\xE2\x84\x83'
+ '\n' + 'For:' + '\n'+ str(convert(n)) + '\n -----' + '\n' + influx + '\n' + telegraf + '\n' + grafana)
```

- Γίνεται έλεγχος του μηνύματος, αν το μήνυμα αντιστοιχεί στο **«/server_info»**, εκτελείται το παραπάνω κομμάτι το οποίο επιστρέφει τις πληροφορίες του εξυπηρετητή **«Θερμοκρασία λειτουργίας, Χρόνος λειτουργίας ,Κατάσταση υπηρεσιών (Influxdb, Telegraf, Grafana)».** Αποθηκεύει στις μεταβλητές temper1 και uptime1 την τιμή της θερμοκρασίας και του χρόνου λειτουργίας (σε ms) αντίστοιχα. Έπειτα μέσω της εντολής **stat1 = os.system('systemctl is-active --quiet <u>όνομα_υπηρεσίας.service</u>)**, ελέγχει την κατάσταση των υπηρεσιών Influxdb, Telegraf και Grafana και επιστρέφει ανάλογο μήνυμα, σε περίπτωση που η υπηρεσία είναι ενεργεί επιστρέφει το 0, ενώ σε περίπτωση που είναι ανενεργή το 768. Τέλος, μέσω της εντολής **bot.sendMessage(chat_id, '\xF0\x9F\x92\xBB SERVER is running at: ' + str(temper) + '\xE2\x84\x83' + '\n' + 'For:' + '\n'+ str(convert(n)) + '\n -----' + '\n' + influx + '\n' + telegraf + '\n' + grafana)** το bot επιστρέφει τα παραπάνω δεδομένα, όσον αφορά τον χρόνο λειτουργίας καλεί την συνάρτηση **convert** η οποία μετατρέπει τα ms σε Μέρες , ώρες, λεπτά και δευτερόλεπτα.

```
elif command == '/last_values':
    results = client.query('SELECT last("payload_fields_Air_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
    time = results.raw['series'][0]['values'][0][0]
    timee = str(time)
    timee.split("T")
    print ('Last Temperature received: %s ' % (time))
    time1, time2 = timee.split("T")
    time2.split(".")
    timee1, timee2 = time2.split(".")
    timee1.split(":")
    timeone1 = timee1.split(":")
    timehour = int(timeone1[0]) + 2
    timehour = str(timehour) + ':' + timeone1[1]
    temperature = results.raw['series'][0]['values'][0][1]
    temperature1 = '\xE2\x96\xAA Air temperature: ' + str(temperature) + '\xE2\x84\x83'
    print ('Last Temperature received: %s , at: %s %s' % (temperature1, timee1, time1))
    results = client.query('SELECT last("payload_fields_Air_Humidity") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
    humidity = results.raw['series'][0]['values'][0][1]
    humidity1 = '\xE2\x96\xAA Air humidity: ' + str(humidity) + '\x25H'
    results = client.query('SELECT last("payload_fields_Air_Pressure") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
    pressure = results.raw['series'][0]['values'][0][1]
    pressure1 = '\xE2\x96\xAA Air pressure: ' + str(pressure) + 'hPa'
    results = client.query('SELECT last("payload_fields_Ground_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
    g_temperature = results.raw['series'][0]['values'][0][1]
    g_temperature1 = '\xE2\x96\xAA Ground temperature: ' + str(g_temperature) + '\xE2\x84\x83'
    results = client.query('SELECT last("payload_fields_Ground_Moisture") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
    g_moisture = results.raw['series'][0]['values'][0][1]
    g_moisture1 = '\xE2\x96\xAA Ground moisture: ' + str(g_moisture) + '\x25'
    bot.sendMessage(chat_id, 'Last values received at: ' + '\n' + '\xF0\x9F\x95\x90 ' + str(timehour) + '\n' + '\xF0\x9F\x93\x86 ' + str(time1) + '\n -----' + '\n' + temperature1 + '\n' + humidity1 + '\n' + pressure1 + '\n' + g_temperature1 + '\n' + g_moisture1)
```

- Στην επόμενη **elif** γίνεται έλεγχος αν το μήνυμα αντιστοιχεί στο **«/last_values»** και εκτελείται το παραπάνω κομμάτι το οποίο επιστρέφει τις τελευταίες μετρήσεις που καταγράφηκαν στην βάση δεδομένων καθώς και την ακριβή στιγμή της καταγραφής.
  Μέσω του query **results = client.query('SELECT last("payload_fields_Air_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')** αποθηκεύεται στην results το object του πεδίου **Air_Temperature**, έπειτα μέσω της εντολής **time = results.raw['series'][0] ['values'][0][0**] αποθηκεύεται στην **time** η ακριβής ώρα της τελευταίας μέτρησης που έχει καταγραφεί την τελευταία μέρα. Με τα αντίστοιχα Queries αντλούνται τα δεδομένα όλων των αισθητήρων και αποστέλλονται από το bot στον τελικό χρήστη.

```
    elif command == '/max_values':
        results = client.query('SELECT max("payload_fields_Air_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        temperature = results.raw['series'][0]['values'][0][1]
        temperature1 = '\xE2\x96\xAA Maximum air temperature: ' + str(temperature) + '\xE2\x84\x83'
        results = client.query('SELECT max("payload_fields_Air_Humidity") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        humidity = results.raw['series'][0]['values'][0][1]
        humidity1 = '\xE2\x96\xAA Maximum air humidity: ' + str(humidity) + '\x25H'
        results = client.query('SELECT max("payload_fields_Air_Pressure") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        pressure = results.raw['series'][0]['values'][0][1]
        pressure1 = '\xE2\x96\xAA Maximum air pressure: ' + str(pressure) + 'hPa'
        results = client.query('SELECT max("payload_fields_Ground_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        g_temperature = results.raw['series'][0]['values'][0][1]
        g_temperature1 = '\xE2\x96\xAA Maximum ground temperature: ' + str(g_temperature) + '\xE2\x84\x83'
        results = client.query('SELECT max("payload_fields_Ground_Moisture") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        g_moisture = results.raw['series'][0]['values'][0][1]
        g_moisture1 = '\xE2\x96\xAA Maximum ground moisture: ' + str(g_moisture) + '\x25'
        bot.sendMessage(chat_id, 'Todays maximum values' + '\n -----' + '\n' + temperature1 + '\n' + humidity1 + '\n' + pressure1 + '\n' + g_temperature1 + '\n' + g_moisture1  )

    elif command == '/min_values':
        results = client.query('SELECT min("payload_fields_Air_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        temperature = results.raw['series'][0]['values'][0][1]
        temperature1 = '\xE2\x96\xAA Minimum air temperature: ' + str(temperature) + '\xE2\x84\x83'
        results = client.query('SELECT min("payload_fields_Air_Humidity") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        humidity = results.raw['series'][0]['values'][0][1]
        humidity1 = '\xE2\x96\xAA Minimum air humidity: ' + str(humidity) + '\x25H'
        results = client.query('SELECT min("payload_fields_Air_Pressure") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        pressure = results.raw['series'][0]['values'][0][1]
        pressure1 = '\xE2\x96\xAA Minimum air pressure: ' + str(pressure) + 'hPa'
        results = client.query('SELECT min("payload_fields_Ground_Temperature") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        g_temperature = results.raw['series'][0]['values'][0][1]
        g_temperature1 = '\xE2\x96\xAA Minimum ground temperature: ' + str(g_temperature) + '\xE2\x84\x83'
        results = client.query('SELECT min("payload_fields_Ground_Moisture") FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')
        g_moisture = results.raw['series'][0]['values'][0][1]
        g_moisture1 = '\xE2\x96\xAA Minimum ground moisture: ' + str(g_moisture) + '\x25'
```

```
    bot.sendMessage(chat_id, 'Todays minimum values' + '\n -----' + '\n' + temperature1 + '\n' + humidity1 + '\n' + pressure1 + '\n' + g_temperature1 + '\n' + g_moisture1  )
```

- Ακολουθεί έλεγχος αν το μήνυμα αντιστοιχεί στο **«/max_values»** ή **«/min_values»** και εκτελείται κώδικας όπου επιστρέφει τις ανώτερες και κατώτερες ημερήσιες τιμές που έχουν καταγραφεί. Οι τιμές αποθηκεύονται στην results με το παρακάτω query: **results = client.query('SELECT min***(στην περίπτωση της ελάχιστης τιμής)* ή **max***(στην περίπτωση της μέγιστης)* **("payload_fields_"**Πεδίο μετρήσεων"**FROM "autogen"."mqtt_consumer" WHERE time > now() -1d ')** και αποστέλλονται στον τελικό χρήστη με τη χρήση της **bot.sendMessage()**.

```
elif command == '/daily_air_temp':
        urllib.urlretrieve('http://192.168.1.48:3000/render/d/loraNode/lora-node_daily?orgId=1&viewPanel=4' , 'air_temp.png')
        bot.sendPhoto(chat_id, photo=open('/home/pi/air_temp.png'))
elif command == '/daily_air_hum':
    urllib.urlretrieve('http://192.168.1.48:3000/render/d/ loraNode /lora-node_daily?orgId=1&viewPanel=7' , 'air_hum.png')
    bot.sendPhoto(chat_id, photo=open('/home/pi/air_hum.png'))
 elif command == '/daily_air_press':
    urllib.urlretrieve('http://192.168.1.48:3000/render/d/ loraNode /lora-node_daily?orgId=1&viewPanel=9' , 'air_press.png')
    bot.sendPhoto(chat_id, photo=open('/home/pi/air_press.png'))
elif command == '/daily_grnd_temp':
    urllib.urlretrieve('http://192.168.1.48:3000/render/d/ loraNode /lora-node_daily?orgId=1&viewPanel=11' , 'grnd_temp.png')
    bot.sendPhoto(chat_id, photo=open('/home/pi/grnd_temp.png'))
elif command == '/daily_grnd_mois':
    urllib.urlretrieve('http://192.168.1.48:3000/render/d/ loraNode /lora-node_daily?orgId=1&viewPanel=13' , 'grnd_mois.png')
    bot.sendPhoto(chat_id, photo=open('/home/pi/grnd_mois.png'))
else :
    bot.sendMessage(chat_id, 'Not recognised command please try again. \nVersion 1.0 \xC2\xA9')
```

- Τέλος ακολουθεί έλεγχος για τις εντολές **/daily_air_temp, /daily_air_hum, /daily_air_press, /daily_grnd_temp** και **/daily_grnd_mois** οι οποίες επιστρέφουν τα γραφήματα της τάσης του κάθε μεγέθους για το τελευταίο εικοσιτετράωρο. Επίσης στο τέλος γίνεται έλεγχος **else** για την περίπτωση που το μήνυμα που έλαβε το bot δεν αντιστοιχεί σε καμία από τις παραπάνω περιπτώσεις και αποστέλλεται κατάλληλο μήνυμα.

  Όσον αφορά την αποστολή των γραφημάτων, γίνεται rendering του εκάστοτε grafana panel και αποθηκεύεται σε αρχείο .png μέσω της εντολής:
  urllib.urlretrieve('http://192.168.1.48:3000/render/d/loraNode/lora-node_daily?orgId=1&viewPanel="αριθμός PANEL"' , '»Όνομα αρχείου».png')
  Αξίζει να σημειωθεί πως για κάθε νέα εκτέλεση της ίδια εντολής **urllib** το αρχείο που είχε δημιουργηθεί προηγουμένως αντικαθίσταται από νέο.

Το telegram bot αποστέλλει το αρχείο .png του γραφήματος μέσω της εντολής:
bot.sendPhoto(chat_id, photo=open('/home/pi/»όνομα_αρχείου».**png'))**

Το συγκεκριμένο πρόγραμμα αποτελεί ένα script γραμμένο σε Python, το οποίο εκτελείται με τη χρήση της εντολής **python Lora_server.py,** ωστόσο κρίθηκε σκόπιμο να εκτελείται σαν υπηρεσία στον εξυπηρετητή. Για το σκοπό αυτό ακολουθήθηκε η παρακάτω διαδικασία:

Αρχικά έγινε εγκατάσταση του πακέτου python-influxdb, με την εντολή sudo apt-get install python-influxdb.

```
cd /lib/systemd/system
sudo nano Lora_server.service
      [Unit]
      Description=Lora services
       After=multi-user.target

      [Service]
       Type=idle
       ExecStart=/usr/bin/python /home/pi/Lora_server.py
       Restart=on-abort

       [Install]
       WantedBy=multi-user.target

sudo chmod 644 /lib/systemd/system/ Lora_server.service
chmod +x /home/pi/ Lora_server.py
sudo systemctl daemon-reload
sudo systemctl start Lora_server.service
```

Εντολή	Λειτουργία
`systemctl start <όνομα_υπηρεσίας>.service`	Εκκίνηση υπηρεσίας
`systemctl stop <όνομα_υπηρεσίας>.service`	Τερματισμός υπηρεσίας
`systemctl reload <όνομα_υπηρεσίας>.service`	Επανεκκίνηση υπηρεσίας
`systemctl status <όνομα_υπηρεσίας>.service`	Τρέχουσα κατάσταση υπηρεσίας
`cd –"~" –"/" –"όνομα καταλόγου"`	Πλοήγηση στον αρχικό, τον root ή συγκεκριμένο κατάλογο
`nano <όνομα αρχείου>`	Επεξεργασία αρχείου με τη χρήση του επεξεργαστή κειμένου nano
`chmod <-παράμετρος, κατάλογος, όνομα αρχείου>`	Προσδιορισμός δικαιωμάτων εκτέλεσης
`sudo <εντολή>`	Εκτέλεση εντολής με δικαιώματα διαχειριστή

Πίνακας 15: Σύνοψη εντολών linux

3.6.4 Προσομοίωση χρήσης της υπηρεσίας Lora_server

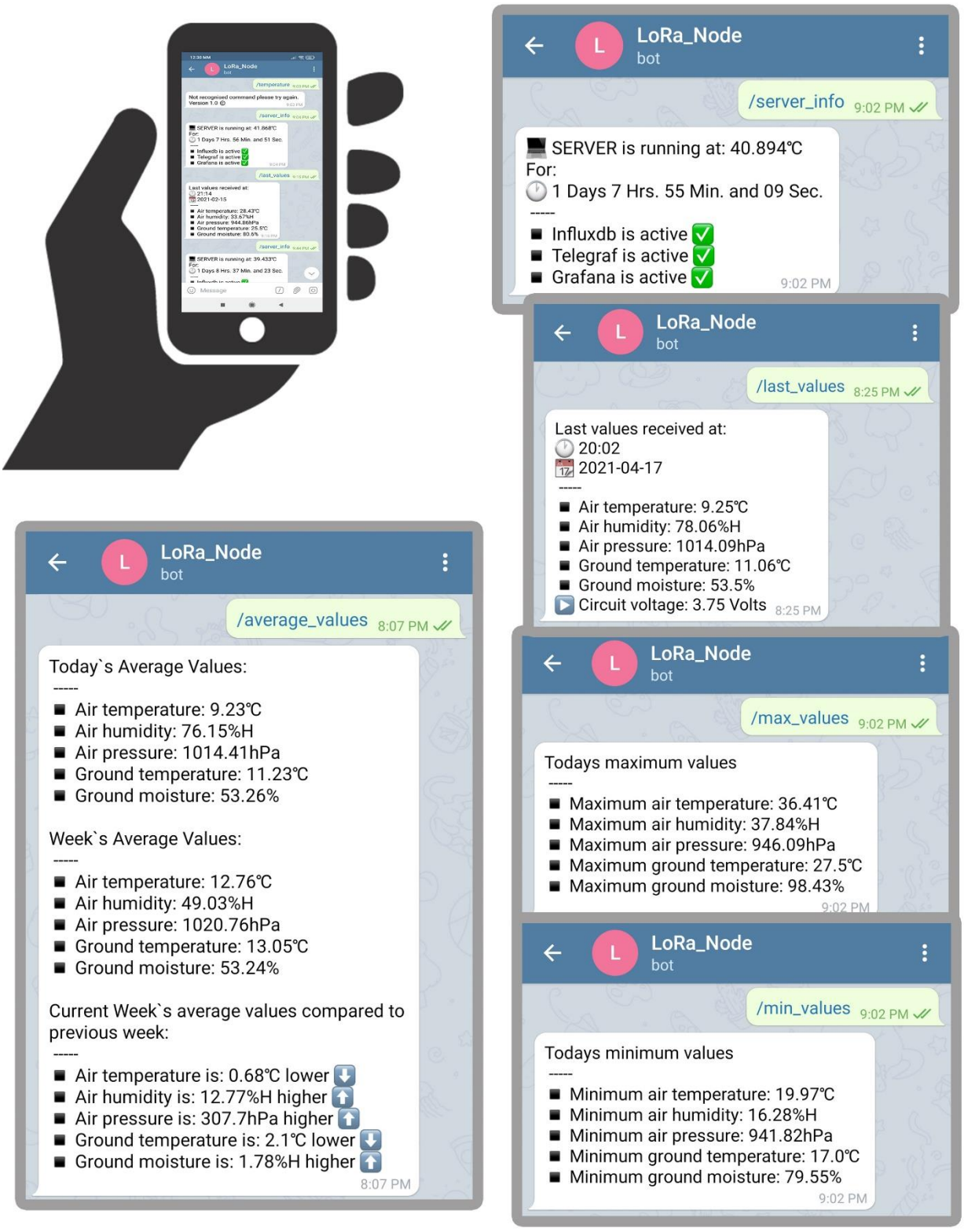
Πραγματοποιήθηκε καθορισμός εντολών για χρήση με το συγκεκριμένο bot. Μέσω του botfather, δόθηκε η εντολή **/setcommands** και έπειτα δόθηκαν όλες οι εντολές με τη μορφή : **εντολή - Περιγραφή εντολής.**

Από τη στιγμή που πλέον το script Lora_server εκτελείται σαν υπηρεσία, είναι σε θέση να ξεκινά τη λειτουργία κατά την εκκίνηση του εξυπηρετητή. Το telegram bot μπορεί να αλληλοεπιδρά με τον τελικό χρήστη σύμφωνα με τις εντολές που λαμβάνει. Ακολουθεί προσομοίωση χρήσης.

Εικόνα 68: Αποθηκευμένες εντολές Telegram bot

Στην περίπτωση που η εντολή δεν αντιστοιχεί σε κάποια από τις παραπάνω και δεν αναγνωρίζεται από το Telegram Bot απαντά με κατάλληλο μήνυμα

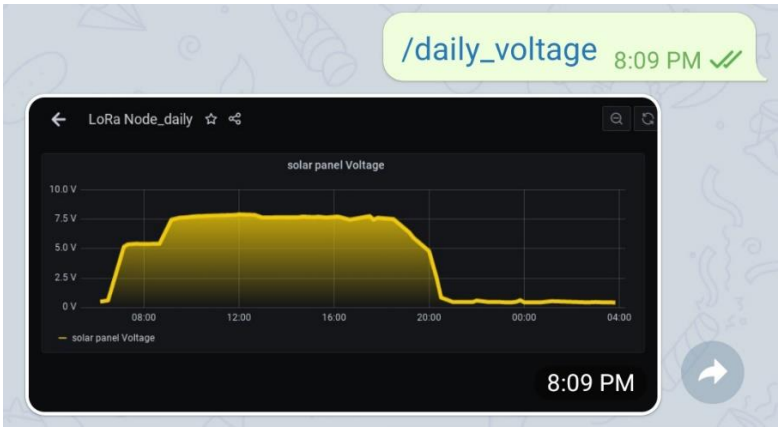
Εικόνα 69: Μη αναγνωρισμένο μήνυμα



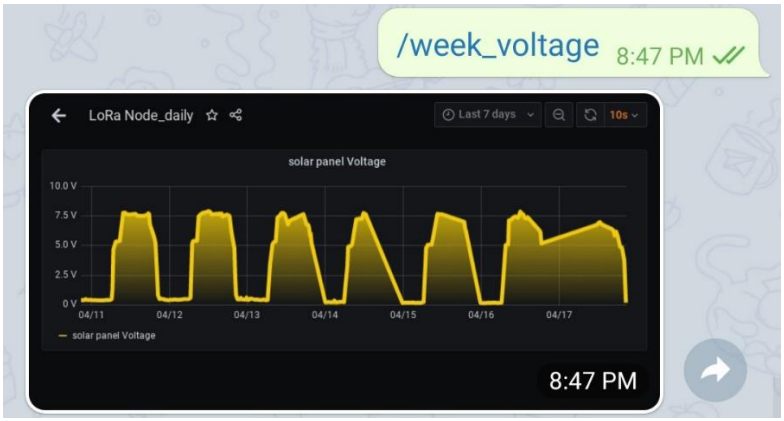
Εικόνα 70: Προσομοίωση χρήσης εντολών /server_info, /last_values, /max_values, /min_values, /average_values

Εικόνα 71: Προσομοίωση χρήσης εντολών /daily_air_temp, /daily_air_hum, /daily_air_press, /daily_grnd_temp, /daily_grnd_mois

Όσον αφορά την τάση του ηλιακού κυκλώματος τροφοδοσίας ο χρήστης έχει τη δυνατότητα να παρατηρήσει το ημερήσιο και το εβδομαδιαίο διάγραμμα της πορείας του.

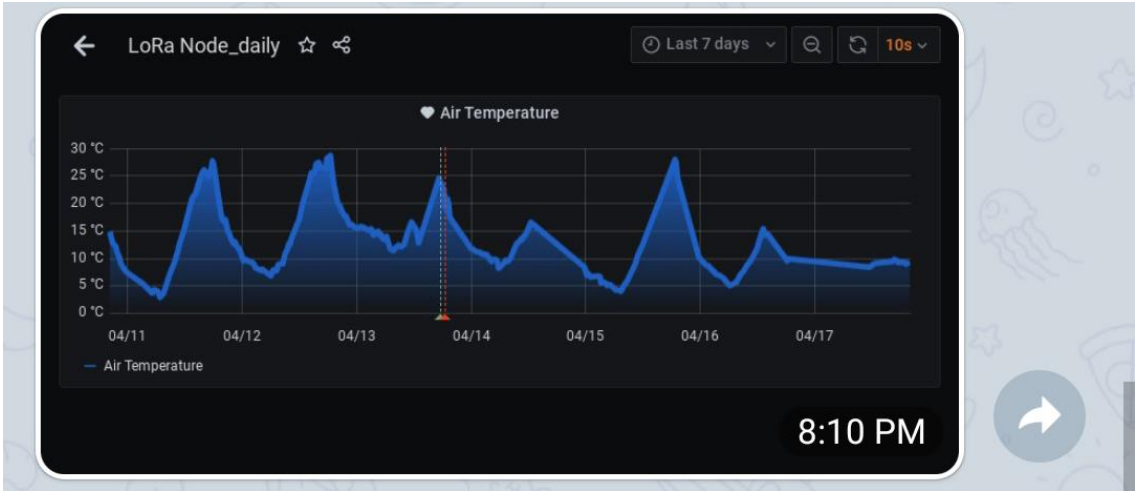


Εικόνα 72: Προσομοίωση χρήσης εντολής /daily_voltage



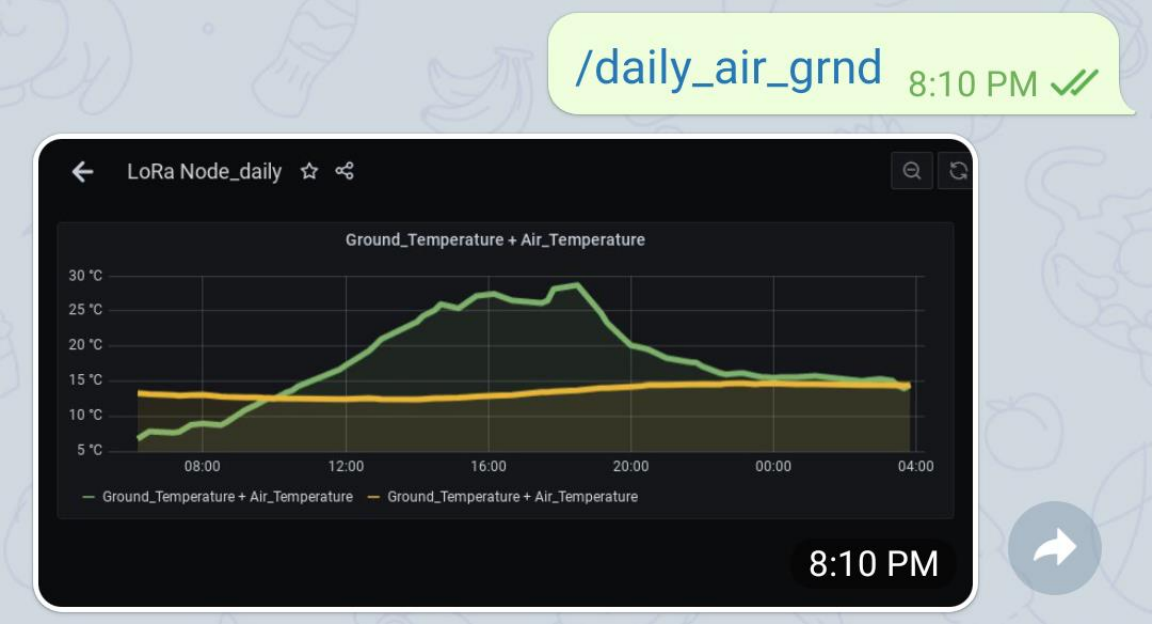
Εικόνα 73: Προσομοίωση χρήσης εντολής /week_voltage

Ο χρήστης δύναται να παρατηρήσει την πορεία της ατμοσφαιρικής θερμοκρασίας σε εβδομαδιαίο επίπεδο καθώς και την πορεία της σε σχέση με τη θερμοκρασία εδάφους τόσο σε εβδομαδιαίο όσο και σε ημερήσιο επίπεδο.



Εικόνα 74: Προσομοίωση χρήσης εντολής /week_air_temp

Εικόνα 75: Προσομοίωση χρήσης εντολής /week_air_grnd

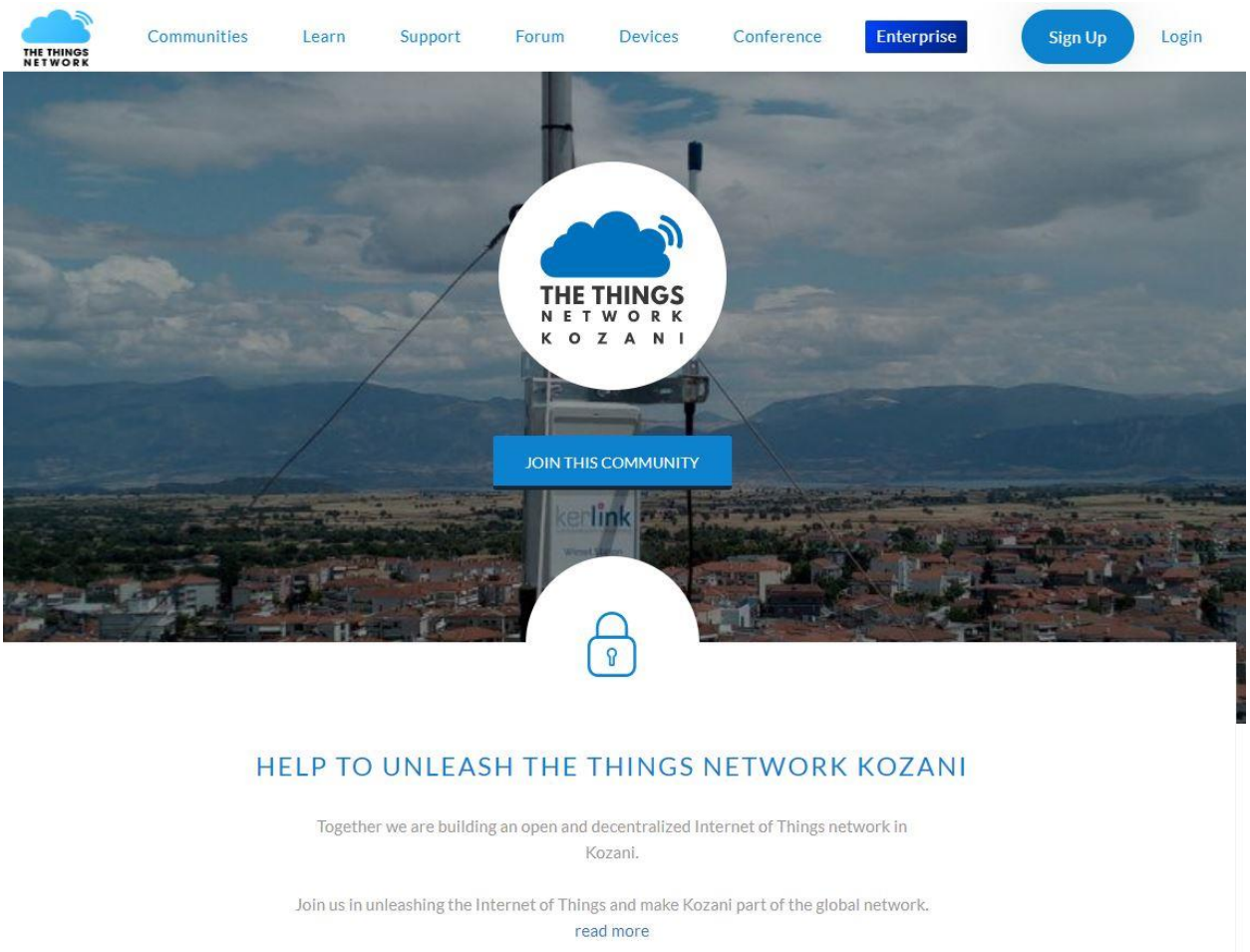


Εικόνα 76: Προσομοίωση χρήσης εντολής /daily_air_grnd

Κεφάλαιο 4ο - Εφαρμογή

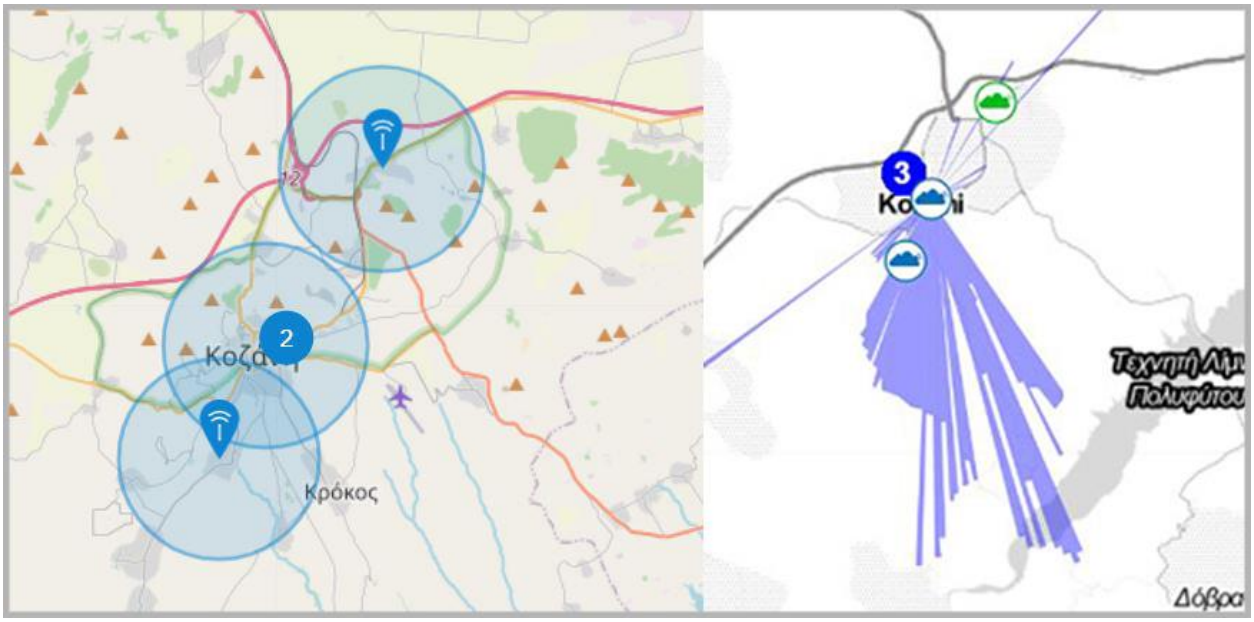
4.1 Εγκατάσταση του αισθητήριου κόμβου

Ο αισθητήριος κόμβος εγκαταστάθηκε σε αγροτική έκταση στην περιοχή το δήμου Κοζάνης και όπως έχει ήδη αναφερθεί η κάλυψη LoRa επιτεύχθηκε μέσω του δικτύου που έχει υλοποιήσει το Πανεπιστήμιο Δυτικής Μακεδονίας. Πρόσφατα ξεκίνησε μια προσπάθεια διεύρυνσης του δικτύου μέσα από την κοινότητα **«THE THINGS NETWORK KOZANI»** από το τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, παρόλα αυτά το δίκτυο προσφέρεται σε κάθε πολίτη που θέλει να συνδέσει τις IoT συσκευές του.



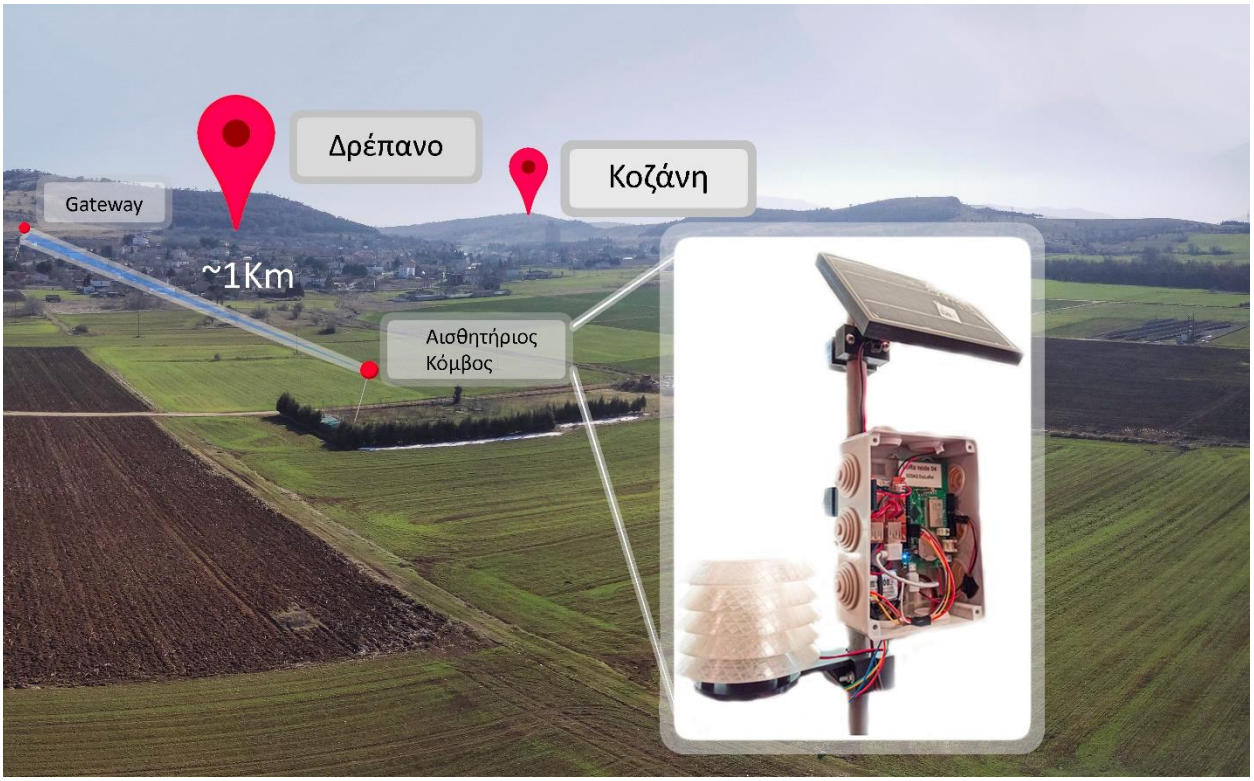
Εικόνα 77: Κοινότητα TTN Kozani

Ακολουθεί η τρέχουσα κάλυψη που παρέχει το δίκτυο LoRa στην περιοχή αποτυπωμένη σε χάρτες.



Εικόνα 78: Κάλυψη LoRa

Όσον αφορά την εγκατάσταση του αισθητήριου κόμβου, οι αισθητήρες εδάφους (θερμοκρασίας και υγρασίας) αρχικά τοποθετήθηκαν σε βάθος περίπου 15 εκατοστών και αφού συνδέθηκε στο εξωτερικό κύκλωμα τροφοδοσίας μέσω του ηλιακού πάνελ ξεκίνησε να αποστέλλει μετρήσεις.

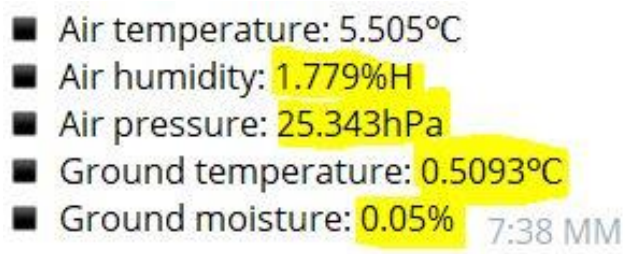


Εικόνα 79: Αισθητήριος κόμβος σε πραγματικές συνθήκες

4.2 Εντοπισμός σφαλμάτων

4.2.1 Διόρθωση κώδικα Arduino IDE

Μετά την εγκατάσταση του αισθητήριου κόμβου και τη λήψη των πρώτων μετρήσεων αντιμετωπίστηκαν τα πρώτα προβλήματα εξαιτίας του λανθασμένου προγραμματισμού. Πιο συγκεκριμένα, τα δεδομένα μετρήσεων που κατέφθαναν έδειχναν να μην ανταποκρίνονται στις πραγματικές καιρικές συνθήκες.



Εικόνα 80: Λήψη λανθασμένων τιμών

Η μόνη τιμή που έδειχνε να ανταποκρίνεται στις τρέχουσες καιρικές συνθήκες ήταν αυτή της ατμοσφαιρικής θερμοκρασίας, ενώ όλες οι υπόλοιπες απέκλιναν σημαντικά. Μάλιστα η ατμοσφαιρική πίεση είχε τιμή κοντά στα 25 hPa δεδομένου ότι στην επιφάνεια της θάλασσας είναι στα 1013 hPa. Για την αντιμετώπιση του προβλήματος έγινε παρακολούθηση της πορείας των δεδομένων. Αρχικά ελέγχθηκαν όλοι οι αισθητήρες και επιβεβαιώθηκε η σωστή λειτουργία τους. Έπειτα έγινε έλεγχος του μηνύματος “payload” που καταφθάνει στο TTN, παρατηρήθηκε πως ενώ ο αισθητήριος κόμβος καταγράφει σωστές τιμές (παρακολούθηση μέσω του serial monitor του Arduino IDE) το payload που φτάνει στο TTN δεν ανταποκρίνεται σε αυτές τις τιμές. Σε αυτό το στάδιο έγινε ο προσδιορισμός του προβλήματος, κατά τη διάρκεια προγραμματισμού του αισθητήριου κόμβου στο Κεφάλαιο 2 έγινε το σημαντικό λάθος να μην ληφθεί υπόψιν η πιθανή μεταβολή της κάθε μέτρησης, δηλαδή ότι η θερμοκρασία μπορεί να είναι είτε διψήφιος είτε μονοψήφιος αριθμός. Επομένως κρίθηκε αναγκαίο να προσδιοριστεί το εύρος μέτρησης του κάθε επιμέρους αισθητήρα και να διορθωθεί ο κώδικας ώστε το payload να είναι πάντοτε σταθερό.

Προσδιορίστηκαν οι μέγιστες και ελάχιστες τιμές της κάθε μέτρηση ως εξής: Ατμοσφαιρική Θερμοκρασία -99.99°C έως +99.99°C, Ατμοσφαιρική Υγρασία 0.00% έως 100.00%, Ατμοσφαιρική Πίεση 900.00hPa έως 1200.00hPa, θερμοκρασία εδάφους -99.99°C έως +99.99°C, Υγρασία εδάφους 0.00% έως 100.00%. Επίσης κρίθηκε σκόπιμο η θερμοκρασίες να μετατρέπονται σε Kelvin (173.16K έως 373.14K) προτού σταλούν και έπειτα να μετατρέπονται ξανά στη κλίμακα Κελσίου από την συνάρτηση του Decoder στο TTN για την αποφυγή αποστολής αρνητικών αριθμών. Τέλος λαμβάνοντας υπόψιν τις παραπάνω τιμές ορίστηκε ως μέγιστο μέγεθος του μηνύματος payload τα 34Byte και διορθώθηκε το αντίστοιχο κομμάτι κώδικα μέσω του Arduino IDE.

Οι αλλαγές έγιναν στην συνάρτηση getValues() ως εξής:

```
String getValues()
{
…..

airtemp = airtemp + 273.15;

grtemp = grtemp + 273.15;
```

- Μετατροπή των τιμών της θερμοκρασίας αέρα και εδάφους σε Kelvin προσθέτοντας το 273,15.

```
 if (airhum <= 9.99)
  {
    airhum1 = "0" + grmois1;
  }

  airpres1 = airpres;
    if (airpres <= 999.99)
  {
    airpres1 = "0" + airpres1;
  }

 if (grmois <= 9.99)
  {
    grmois1 = "0" + grmois1;
  }
```

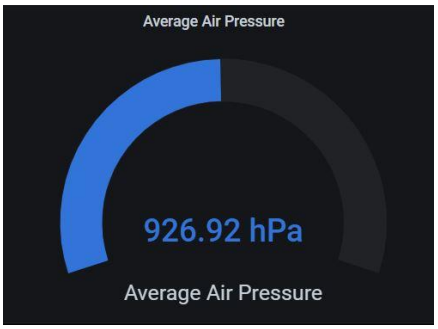
- Έλεγχος των τιμών που επιστρέφονται από τον αισθητήρα BME280 για τις μετρήσεις υγρασίας και ατμοσφαιρικής πίεσης καθώς και του αισθητήρια υγρασίας εδάφους. Στην περίπτωση που οι τιμές είναι μονοψήφιος αριθμός ή τριψήφιος για την ατμοσφαιρική πίεση προστίθεται το μηδέν ώστε να καταλαμβάνουν το μέγιστο μήκος σε byte κρατώντας έτσι το payload σταθερό. Δεν υπάρχει λόγος ελέγχου των θερμοκρασιών εδάφους και αέρα καθώς μετά την μετατροπή τους και τον προσδιορισμό των μέγιστων και ελαχίστων τιμών που έγινε παραπάνω πρόκειται για τριψήφιους αριθμούς δύο δεκαδικών σταθερού μήκους 5 byte ο κάθε ένας.

Ακολούθησε και αντίστοιχη τροποποίηση στην συνάρτηση της αποκωδικοποίησης (decoder) στην κονσόλα της εφαρμογής στο The Things Network. Η τροποποίηση αφορά την εκ νέου μετατροπή των θερμοκρασιών στην κλίμακα Κελσίου, αφαιρώντας απλά αυτή τη φορά τη σταθερά Kelvin.

```
var x = 273.15;

x = parseFloat(x);

result1 = parseFloat(result1) - x ; //Η τιμή της ατμοσφαιρική θερμοκρασίας

result4 = parseFloat(result4) - x ; //Η τιμή της θερμοκρασίας εδάφους
```

4.2.2 Βαθμονόμηση αισθητήρα BME280

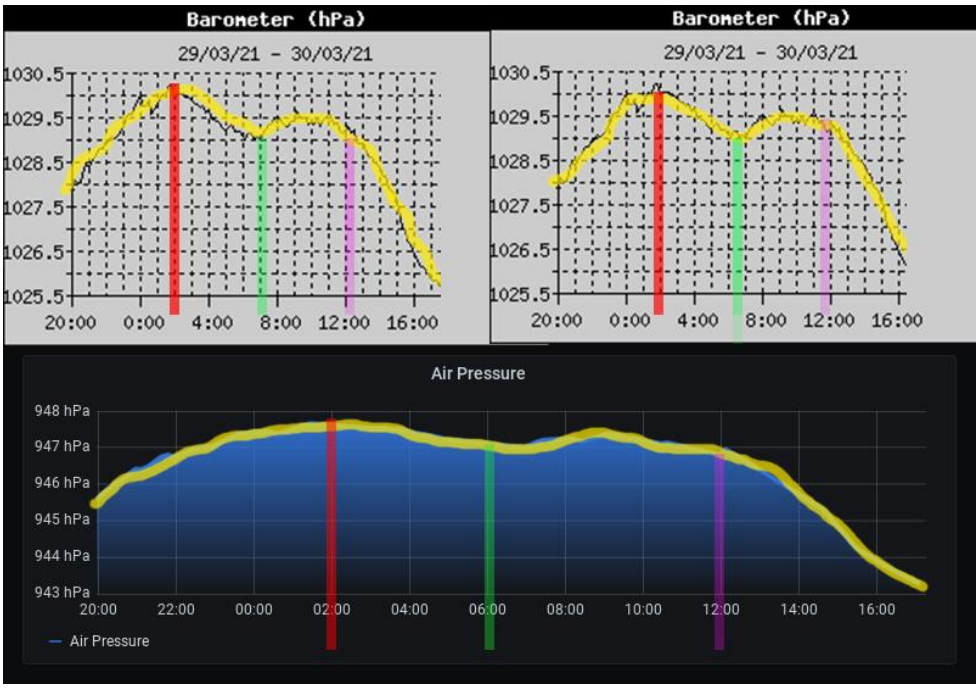
Αφού βεβαιώθηκε η ορθή λειτουργία του αισθητήριου κόμβου, μετά από σύγκριση των μετρήσεων με ανοιχτά δεδομένα μετεωρολογικών σταθμών στην περιοχή παρατηρήθηκε πως όλοι οι επί μέρους αισθητήρες λειτουργούν σωστά εκτός του BME280 και συγκεκριμένα της μέτρησης της βαρομετρικής πίεσης. Σε ένα διάστημα δύο ημερών καταγράφηκε μέση τιμή βαρομετρικής πίεσης **926.92 hPa**, μια τιμή που απέκλινε πολύ από τις μετρήσεις άλλων σταθμών της περιοχής.



Εικόνα 81: Μέση ατμοσφαιρική πίεση

Ακολούθησε καταγραφή των μετρήσεων σε διάστημα ενός εικοσιτετράωρου και προέκυψε το συμπέρασμα πως το ίχνος των μετρήσεων στο διάγραμμα μέτρησης-χρόνου του BME280 είναι παράλληλο με σταθερή διαφορά από αυτό των μετρήσεων άλλων σταθμών της περιοχής.

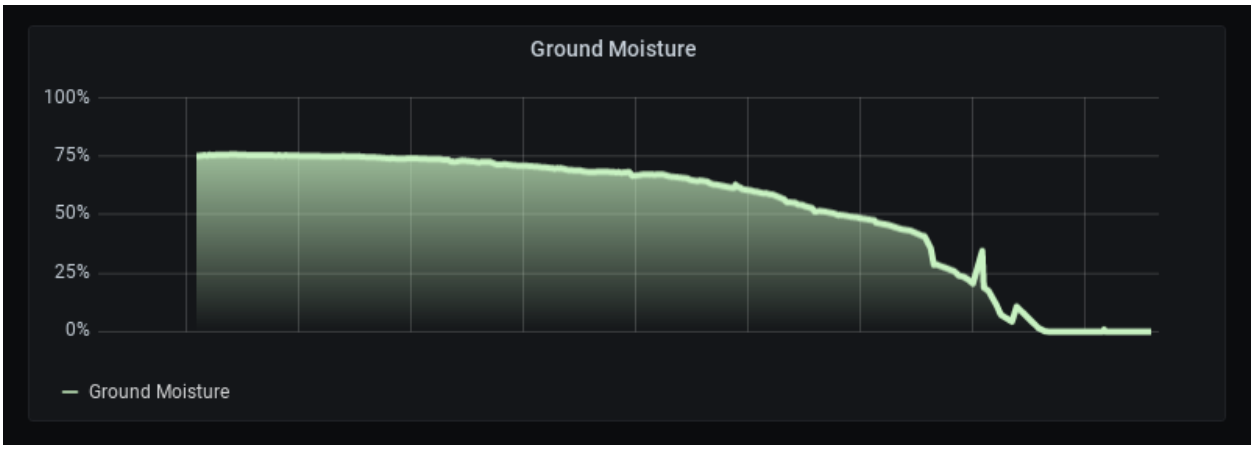
Συγκεκριμένα, χρησιμοποιήθηκαν τα δεδομένα των σταθμών « **Κίτρινη Λίμνη Κοζάνης**» σε υψόμετρο 664μέτρα και «**Κοζάνη**» σε υψόμετρο 768μέτρα ιδιοκτησίας του Εθνικού Αστεροσκοπείου Αθηνών και Δήμου Κοζάνης αντίστοιχα, που προσφέρονται μέσω του www.meteo.gr. Υπολογίστηκε η μέση τιμή απόκλισης στα **82.25hPa** και διορθώθηκε στον κώδικα του αισθητήριου κόμβου μέσω του Arduino IDE προσθέτοντας το 82,54 στην τιμή της βαρομετρικής πίεσης.



Εικόνα 82: Σύγκριση ατμοσφαιρικής πίεσης με open data

4.2.3 Αξιοπιστία αισθητήρα YL-69 YL-39

Μετά από χρήση του αισθητήριου κόμβου σε διάστημα δεκαπέντε ημερών ο αισθητήρας μέτρησης της υγρασίας εδάφους παρουσίασε τα πρώτα προβλήματα. Παρατηρήθηκε εκτεταμένη οξείδωση στο αισθητήριο YL-69.



Εικόνα 83: Γράφημα υγρασίας εδάφους

Εικόνα 84: Οξείδωση αισθητήριου YL-69

Όπως έχει ήδη αναφερθεί προηγουμένως στο κεφάλαιο 2 κατά την παρουσίαση του εξοπλισμού, ο συγκεκριμένος αισθητήρας λειτουργεί μετρώντας την αντίσταση που αντιμετωπίζει το ρεύμα από το ένα άκρο στο άλλο. Πιο συγκεκριμένα εφαρμόζεται το φαινόμενο της ηλεκτρόλυσης κατά το οποίο τα άτομα χαλκού του θετικού άκρου (το οποίο έχει υποστεί τη μεγαλύτερη φθορά) μεταφέρονται μέσω της υγρασίας του χώματος στο αρνητικό.

Το συγκεκριμένο πρόβλημα μπορεί να μειωθεί αλλά όχι να εξαλειφθεί αν δεν εφαρμόζεται συνεχώς τάση στο αισθητήριο παρά μόνο τη στιγμή της μέτρησης. Δεδομένου ότι σκοπός είναι η κατασκευή ενός αξιόπιστου αισθητήριου κόμβου κρίθηκε σκόπιμο να βρεθούν εναλλακτικές λύσεις ώστε να αντικαταστήσουν το YL-69.

4.2.4 Αισθητήρες Capacitive moisture Sensor V2 και Resistance moisture sensor HD-38

Για τους λόγους που αναφέρθηκαν στην προηγούμενη ενότητα αποφασίστηκε να δοκιμαστούν ακόμα δύο διαφορετικοί αισθητήρες υγρασίας εδάφους. Πρόκειται για τον αισθητήρα V2 capacitive sensor ο οποίος λειτουργεί με διαφορετικό τρόπο από τον αρχικό YL-69, και τον αισθητήρα resistance moisture sensor HD-38 , ο οποίος αν και λειτουργεί με παρόμοιο τρόπο του YL-69 υπερτερεί στην ποιότητα του υλικού.

4.2.4.1 Capacitive moisture sensor V2

Το αισθητήριο Capacitive soil moisture sensor V2 χρησιμοποιείται για την ανίχνευση της υγρασίας του εδάφους. Σε αντίθεση με τους κοινούς αισθητήρες ανίχνευσης της αντίστασής του ηλεκτρικού ρεύματος ο συγκεκριμένος εκμεταλλεύεται την ιδιότητα μεταβολής της χωρητικότητας ενός πυκνωτή όταν ανάμεσα στις πλάκες του καταγράφεται υγρασία. Διαθέτει κατάλληλο κύκλωμα το οποίο χρησιμοποιώντας ένα ολοκληρωμένο τσιπ (N555 timer) μετατρέπει τον χρόνο φόρτιση του πυκνωτή σε ψηφιακά σήματα. Ενδείκνυται για επιφανειακές μετρήσεις καθώς έχει εκτεθειμένα τα ηλεκτρονικά του εξαρτήματα, το κυριότερο πλεονέκτημα του αισθητήρα είναι η μεγάλη διάρκεια ζωής του καθώς δεν αντιμετωπίζει προβλήματα οξείδωσης, χρησιμοποιείται σε αυτοματισμούς και μετεωρολογικούς σταθμούς.

Εικόνα 85: Αισθητήρας capacitive moisture sensor V2

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	3.3-5V
Διασύνδεση	Ενός ψηφιακού pin (D0) / Ενός αναλογικού pin (AO)
Αναλογικό σήμα εξόδου	0 – 3V
timer chip	N555
Εύρος υγρασίας	0-100%rh
Ρεύμα λειτουργίας	5mA

Πίνακας 16: : Χαρακτηριστικά capacitive moisture sensor V2

4.2.4.2 Resistance moisture sensor HD-38

Ο συγκεκριμένος αισθητήρας είναι παρόμοιος του αρχικού YL-69 που χρησιμοποιείται για την ανίχνευση της υγρασίας του εδάφους ανιχνεύοντας της αντίσταση του εδάφους εκμεταλλευόμενος του φαινομένου της ηλεκτρόλυσης. Αποτελείται από δυο μέρη , από τον ανιχνευτή-probe, και την ηλεκτρονική πλακέτα (HD-38). Η διαφορά του με τον YL-69 σημειώνεται στο μήκος του ανιχνευτή αλλά και στο υλικό των ηλεκτροδίων, το οποίο τον καθιστά πιο ανθεκτικό στο φαινόμενο της ηλεκτρόλυσης. Όπως ο YL-69 έτσι και αυτός ο αισθητήρας αναμένεται να παρουσιάσει σημάδια οξείδωσης αλλά σε πολύ μεγαλύτερο χρονικό διάστημα.

Εικόνα 86: Resistance moisture sensor HD-38

Τεχνικά Χαρακτηριστικά

Τάση λειτουργίας	3.3-12V
Διασύνδεση	Ενός ψηφιακού pin (D0) / Ενός αναλογικού pin (AO)
Αναλογικό σήμα εξόδου	0 – 3V
Μήκος	7.5 cm
Comparator Chip	LM393
Εύρος υγρασίας	0-100%rh
Ρεύμα λειτουργίας	20mA
Εφαρμογή	Επιφανειακή/σε βάθος

Πίνακας 17: : Χαρακτηριστικά resistance sensor HD-38

4.2.5 Σύγκριση δεδομένων αισθητήρων υγρασίας εδάφους

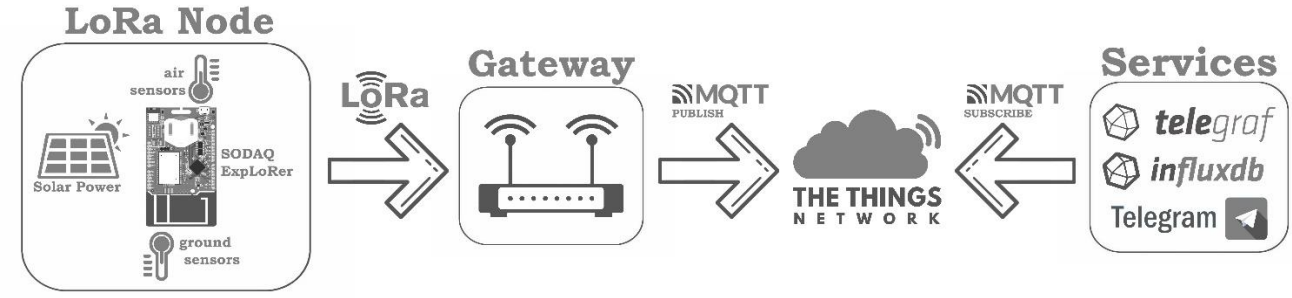
Οι επιπλέον αισθητήρες συνδέθηκαν στον αισθητήριο κόμβο και αφού ο παλιός ανιχνευτής YL-69 που είχε υποστεί οξείδωση αντικαταστάθηκε με καινούριο, τοποθετήθηκαν μαζί στο ίδιο σημείο για τη σύγκριση των μετρήσεών τους. Μετά από λήψη δεδομένων για δέκα ημέρες σημειώθηκαν σημαντικά συμπεράσματα τα οποία αποτυπώνονται και στο παρακάτω διάγραμμα, με μπλε χρώμα είναι η πορεία των μετρήσεων του YL-69, με ροζ του Capacitive V2 και με κίτρινο του Resistance HD-38. Αυτή τη φορά ο YL-69 άρχισε να σημειώνει δυσλειτουργία από την τρίτη μέρα ενώ από την τέταρτη οξειδώθηκε πλήρως. Ο Capacitive V2 και ο Resistance HD-38 συνέχισαν την ομαλή λειτουργία τους με μέση διαφορά **35.18%**, η διαφορά που καταγράφηκε αν και είναι σημαντική πιθανότατα οφείλεται σε λανθασμένες προκαθορισμένες τιμές που δίνονται από τον κατασκευαστή και μπορεί να διορθωθεί με κατάλληλη βαθμονόμηση, καθώς η πορεία των δεδομένων του ενός είναι παράλληλη με την πορεία του δευτέρου. Αξίζει να σημειωθεί όμως πως ο αισθητήρας Resistance HD-38 παρουσίασε καλύτερη ανταπόκριση στην μεταβολή της υγρασίας του εδάφους ενώ ο Capacitive V2 υπερτερεί στην ανθεκτικότητα, τελικά ο YL-69 κρίθηκε ακατάλληλος για χρήση στον τελικό αισθητήριο κόμβο.

Εικόνα 87: Σύγκριση υγρασίας εδάφους και από τους τρεις αισθητήρες

4.3 Τελική μορφή αισθητήριου κόμβου LoRa

Μετά από τις απαραίτητες διορθώσεις τόσο σε αλλαγή εξοπλισμού όσο και σε αποσφαλμάτωση του κυρίως προγράμματος ο αισθητήριος κόμβος έλαβε την τελική του μορφή. Όσο για την επιλογή του αισθητήρα υγρασίας εδάφους, αποφασίστηκε να χρησιμοποιηθεί ο νέος Resistance moisture sensor με την πλακέτα HD-38. Ακολουθεί φωτογραφία του αισθητήριου κόμβου σε πραγματικές συνθήκες καθώς και σχεδιάγραμμα ροής των δεδομένων από τη λήψη των μετρήσεων έως τον τελικό χρήστη.

Εικόνα 88: Τελικός αισθητήριος κόμβος



Εικόνα 89: Ροή δεδομένων αισθητήριου κόμβου προς τις υπηρεσίες

Κεφάλαιο 5ο - Συσχέτιση δεδομένων ανοιχτών πηγών με αυτά του αισθητήριου κόμβου LoRa

5.1 Πηγές Ανοιχτών δεδομένων API's

Στην τελική προσπάθεια λήψης ενός συμπεράσματος για την αξιοπιστία των μετεωρολογικών δεδομένων που παρέχει ο αισθητήριος κόμβος, αποφασίστηκε να πραγματοποιηθεί συσχέτιση των δεδομένων του με αυτά που παρέχουν πηγές ανοιχτών μετεωρολογικών δεδομένων στην περιοχή. Παρακάτω ακολουθεί η αναλυτική περιγραφή των API's που επιλέχθηκαν, της διαδικασίας αποθήκευσής των δεδομένων τους, καθώς και της μεθόδου συσχέτισής τους. Η συλλογή των μετεωρολογικών δεδομένων πραγματοποιήθηκε μέσω της πλατφόρμας του OpenWeatherMap, του Weather Underground και του AccuWeather.

5.1.1 OpenWeatherMap

Εικόνα 90: Λογότυπο OpenWeatherMap

Η δημοφιλής πλατφόρμα του OpenWeatherMap παρέχει μετεωρολογικά δεδομένα σε διάφορες περιοχές σε όλο τον κόσμο. Τα δεδομένα που συλλέγει βασίζονται κυρίως στην συνεισφορά της κοινότητας μέσω προσωπικών μετεωρολογικών σταθμών, καθώς και σε σταθμούς METAR (μετεωρολογικοί σταθμοί αναφοράς καιρού για αεροπορική χρήση) αλλά και σε σταθμούς που ανήκουν στην ίδια την πλατφόρμα. Όσον αφορά τις μετεωρολογικές προβλέψεις, βασίζεται σε μοντέλα αριθμητικής πρόβλεψης καιρού (Numerical Weather Prediction-NWP) που έχουν αναπτυχθεί από την πλατφόρμα. Εκτός από τις τρέχουσες συνθήκες και τις μετεωρολογικές προβλέψεις, προσφέρει και ειδική έκδοση API δίνοντας τη δυνατότητα στους προγραμματιστές να αξιοποιήσουν τα δεδομένα της για χρήση σε διάφορες εφαρμογές.

Το OpenWeatherMap API προσφέρεται σε διάφορες εκδόσεις καλύπτοντας τις διάφορες ανάγκες που προκύπτουν. Πιο συγκεκριμένα, προσφέρεται σε πέντε εκδόσεις, δωρεάν-Free, Startup, Developer, Professional και Enterprise. Όσον αφορά το One Call API, το οποίο ενσωματώνει πληροφορίες του DarkSky προσφέρεται δωρεάν για 1000 κλήσεις/ημέρα στην έκδοση Free και 2000 κλήσεις/ημέρα στην έκδοση Startup. Ακολουθεί αναλυτικός πίνακας αποτυπώνοντας τις υπηρεσίες που προσφέρει η κάθε έκδοση.

Υπηρεσίες	**Free**	**Startup**	**Developer**	**Professional**	**Enterprise**
	60 κλήσεις/λεπτό	*600 κλήσεις/λεπτό*	*3000 κλήσεις/λεπτό*	*30000 κλήσεις/λεπτό*	*200000 κλήσεις/λεπτό*
	1000000 κλήσεις/μήνα	*10000000 κλήσεις/μήνα*	*100000000 κλήσεις/μήνα*	*1000000000 κλήσεις/μήνα*	*5000000000 κλήσεις/μήνα*
Τρέχουσες καιρικές συνθήκες	✓	✓	✓	✓	✓
Πρόβλεψη επόμενης ώρας/λεπτό	*Μέσω του One Call API	*Μέσω του One Call API	✓	✓	✓
Πρόβλεψη επόμενων 4 ημερών/ ώρα	*Επόμενων 2 ημερών, μέσω του One Call API	*Επόμενων 2 ημερών, μέσω του One Call API	✓	✓	✓
Πρόβλεψη επόμενων 16 ημερών/ ημέρα	*Επόμενων 7 ημερών, μέσω του One Call API	✓	✓	✓	✓
Εθνικές προειδοποιήσεις καιρού	*Μέσω του One Call API	*Μέσω του One Call API	✓	✓	✓
Ιστορικό καιρού 5 ημερών	*Μέσω του One Call API	*Μέσω του One Call API	✓	✓	✓
Κλιματική πρόβλεψη 30 ημερών	✗	✗	✓	✓	✓
Λήψη αρχείων	✗	✗	✗	✓	✓
Καιρικοί χάρτες	*Βασικοί χάρτες	*Βασικοί χάρτες	*Προηγμένοι και χάρτες ιστορικού	✓	✓
Κίνδυνοι οδικού δικτύου	✗	✗	✗	✗	✓
Ατμοσφαιρική ρύπανση	✓	✓	✓	✓	✓
Γεωκωδικοποίηση	✓	✓	✓	✓	✓
Widget καιρού	✓	✓	✓	✓	✓
Διαθεσιμότητα	95%	95%	99.5%	99.5%	99.9%
Κόστος	***ΔΩΡΕΑΝ***	*40$/Μήνα*	*180$/Μήνα*	*470$/Μήνα*	*2000$/Μήνα*

Πίνακας 18: Χαρακτηριστικά εκδόσεων OpenWeatherMap [35] [36]

Για τη χρήση του OpenWeatherMap API διατίθεται αναλυτική σελίδα (documentation page). Μετά από την εγγραφή στην πλατφόρμα και την παροχή ενός κλειδιού (API key), καθίσταται άμεσα εφικτή η δυνατότητα χρήσης της εκάστοτε υπηρεσίας του API. Για την συλλογή των δεδομένων

αποστέλλεται ένα HTTP GET Request στο οποίο έχουν καθοριστεί οι παράμετροι και το μοναδικό API key. Τα δεδομένα επιστρέφονται από προεπιλογή σε μορφή JSON αλλά είναι εφικτή και χρήση της μορφής XML και HTML. Ακολουθεί παράδειγμα κλήσης του API για την συλλογή των τρεχουσών καιρικών συνθηκών στην περιοχή της Κοζάνης καθώς και εξήγηση του JSON αντικειμένου που επιστρέφεται.

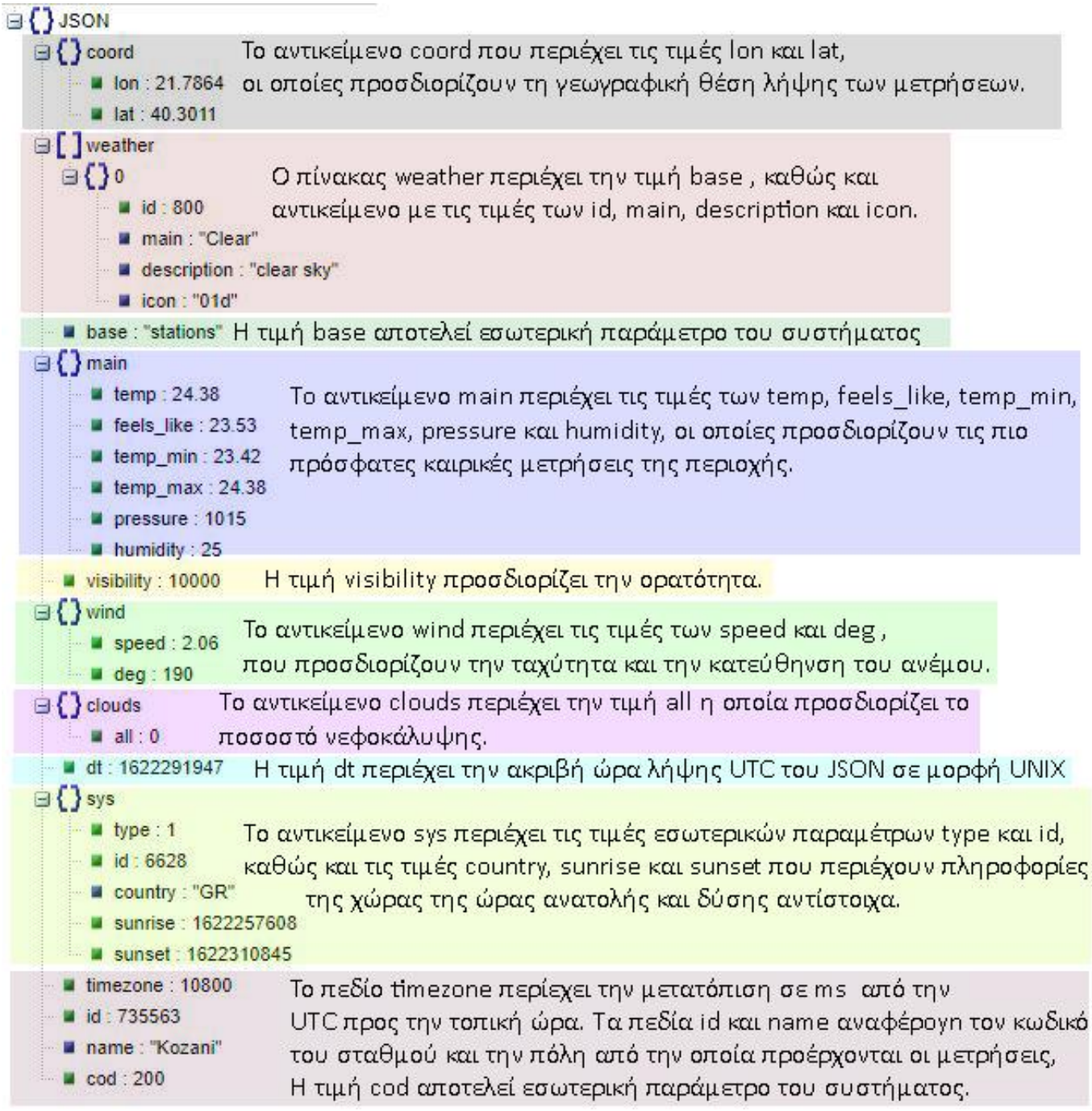
- Σε Linux terminal δίνεται η παρακάτω εντολή (το api μπορεί να κληθεί τόσο από τερματικά Linux/Windows όσο και απευθείας από κάποιον φυλλομετρητή).

  ```
  curl -X GET
  "http://api.openweathermap.org/data/2.5/weather?q=kozani&units=me
  tric&appid={api_key}"
  ```

  Η παράμετρος `q` προσδιορίζει το όνομα της πόλης, η `units` την μονάδα μέτρησης και το `appid` το προσωπικό κλειδί.

Εικόνα 91: Κλήση του API από τερματικό Linux

- Έπειτα επιστρέφεται το JSON που ακολουθεί.



Εικόνα 92: Το JSON αντικείμενο που επιστρέφεται από την κλήση του OpenWeatherMap API [37]

Αναφορικά με την κλήση του API για της τρέχουσες συνθήκες μιας περιοχής εκτός από την κλήση με προσδιορισμό της πόλης υπάρχουν και άλλες δυνατότητες κλήσης.

- Κλήση του API προσδιορίζοντας το `city ID`, που ουσιαστικά δύναται εφικτό να γίνει λήψη δεδομένων από συγκεκριμένο μετεωρολογικό σταθμό.
  `api.openweathermap.org/data/2.5/weather?id={city id}&appid={API key}`

- Κλήση του API με τις γεωγραφικές συντεταγμένες μιας περιοχής μέσω των `lat` και `lon` (γεωγραφικό πλάτος και μήκος).
  `api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}`

- Κλήση του API με χρήση του ταχυδρομικού κώδικα και του κωδικού της χώρας μιας περιοχής μέσω των παραμέτρων `zip code` και `country code`.
  `api.openweathermap.org/data/2.5/weather?zip={zip code},{country code}&appid={API key}`

- Κλήση του API για λήψη δεδομένων πολλαπλών πόλεων εντός ενός ορθογωνίου που καθορίζεται από γεωγραφικές συντεταγμένες μέσω της παραμέτρου {`bbox`} που περιέχει τις άκρες του ορθογωνίου `[lon-left,lat-bottom,lon-right,lat-top,zoom]`.
  `api.openweathermap.org/data/2.5/box/city?bbox={bbox}&appid={API key}`

- Κλήση του API για λήψη δεδομένων πολλαπλών πόλεων εντός ενός οριοθετημένου κύκλου που προσδιορίζεται από το κέντρο και την ακτίνα μέσω των παραμέτρων `lat`, `lon` και `cnt`.
  `api.openweathermap.org/data/2.5/find?lat={lat}&lon={lon}&cnt={cnt}&appid={API key}`

5.1.2 One Call API (OpenWeatherMap)

Το One Call API, είναι μια διεπαφή που διατίθεται από την πλατφόρμα του OpenWeatherMap και ενσωματώνει δεδομένα από την πλατφόρμα του Dark Sky API. Ουσιαστικά προσφέρει με μόνο μια κλήση ενός HTTP Request δεδομένα των τρεχουσών καιρικών συνθηκών μαζί με προβλέψεις για τις επόμενες επτά ημέρες ή ιστορικά δεδομένα καιρού των προηγούμενων πέντε ημερών. Το One Call API δέχεται παρόμοιες παραμέτρους με αυτές του OpenWeatherMap API και όπως έχει ήδη αναφερθεί προηγουμένως προσφέρεται δωρεάν για χίλιες κλήσεις ανά ημέρα στην Free έκδοση και δύο χιλιάδες κλήσεις ανά ημέρα στην έκδοση Startup.

Ακολουθεί παράδειγμα κλήσης του API και λήψη των τρεχουσών καιρικών συνθηκών με πρόγνωση για της επόμενες επτά ημέρες σε ένα ενιαίο JSON.

```
https://api.openweathermap.org/data/2.5/onecall?lat=40.2989955&lon=21.7822009&appid=APPI_KEY
```

Το JSON που επιστρέφεται περιέχει τις τιμές των γεωγραφικών συντεταγμένων που δόθηκαν ως ορίσματα, καθώς και τις τιμές της ζώνης ώρας και μετατόπισης σε ms από την UTC. Έπειτα δημιουργεί το αντικείμενο `current` το οποίο περιέχει τις τιμές με τα τρέχοντα μετεωρολογικά δεδομένα που επικρατούν στην περιοχή. Ο πίνακας minutely περιέχει αντικείμενα από το 0 έως το 60, τα οποία με τη σειρά τους περιέχουν τις τιμές `dt` και `precipitation`, οι οποίες περιλαμβάνουν τις πληροφορίες της ακριβής ώρας σε UNIX και την πρόγνωση για βροχόπτωση το συγκεκριμένο λεπτό. Ο πίνακας `hourly` δημιουργεί αντικείμενα από το 0 που περιέχουν πιο αναλυτικές πληροφορίες που αφορούν την ωριαία πρόγνωση του καιρού, ενώ ο πίνακας `daily` δημιουργεί αντικείμενα από το 0 έως το 7 εμπεριέχοντας ακόμα πιο αναλυτικά δεδομένα που αφορούν την ημερήσια πρόβλεψη για τις επόμενες επτά ημέρες. Τέλος ο πίνακας `alerts` περιέχει πληροφορίες εθνικών καιρικών προειδοποιήσεων.

- {} JSON
  - lat : 40.299
  - lon : 21.7822
  - timezone : "Europe/Athens"
  - timezone_offset : 10800
  - {} current
  - [] minutely
    - {} 0
      - dt : 1622379900
      - precipitation : 0
    - {} .....
  - [] hourly
    - {} 0
    - {} .....
  - [] daily
    - {} 0
    - {} .....
  - [] alerts

Όσον αφορά την κλήση του One Call API για λήψη ιστορικών μετεωρολογικών δεδομένων, δέχεται ως ορίσματα τις γεωγραφικές συντεταγμένες, καθώς και την ακριβή ώρα μια ημέρας σε μορφή UNIX εντός των προηγούμενων πέντε ημερών. Ακολουθεί παράδειγμα κλήσης του API και εξήγηση του JSON που επιστρέφεται.

```
https://api.openweathermap.org/data/2.5/onecall/timemachine?lat=40.2989955&lon=21.7822009&dt=1621952700&appid=API_KEY
```

Το JSON που επιστρέφεται περιέχει και αυτό τις τιμές των γεωγραφικών συντεταγμένων που δόθηκαν ως ορίσματα, τα δεδομένα της ζώνης ώρας, καθώς τα τρέχοντα μετεωρολογικά δεδομένα στο αντικείμενο `current`. Ο πίνακας `hourly` περιέχει αντικείμενα από το 0 έως το 23, από τα οποία το κάθε ένα έχει τα ιστορικά δεδομένα μιας ώρας ενός εικοσιτετράωρου, με πρώτο το αντικείμενο 0 να έχει τα δεδομένα της ώρας που δόθηκε κατά την κλήση του API στην παράμετρο `dt` και τελευταίο το 23 με τα δεδομένα της εικοστής τέταρτης ώρας. Το κάθε αντικείμενο περιλαμβάνει την μέση τιμή της κάθε καιρικής μέτρησης καθώς και γενικότερη περιγραφή των συνθηκών που επικράτησαν την συγκεκριμένη ώρα.

- JSON
  - lat : 40.299
  - lon : 21.7822
  - timezone : "Europe/Athens"
  - timezone_offset : 10800
  - current
  - hourly
    - 0
      - dt : 1621900800
      - temp : 293.35
      - feels_like : 292.83
      - pressure : 1020
      - humidity : 59
      - dew_point : 285.09
      - clouds : 89
      - wind_speed : 1.59
      - wind_deg : 163
      - wind_gust : 1.94
      - weather
    - .....
    - 23

5.1.3 AccuWeather

Εικόνα 93: Λογότυπο AccuWeather

Η AccuWeather είναι αμερικανική εταιρία στον τομέα της μετεωρολογίας που μεταξύ άλλων παρέχει μετεωρολογικά δεδομένα για όλο τον κόσμο μέσω της ομώνυμης πλατφόρμας. Η AccuWeather

αναφέρει πως αντλεί καιρικά δεδομένα από έγκυρες πηγές χωρίς να διευκρινίζει τους συγκεκριμένους μετεωρολογικούς σταθμούς που χρησιμοποιεί, ενώ προβλέψεις που παρέχει βασίζονται σε διάφορα αριθμητικά μοντέλα καιρού. Η πλατφόρμα εκτός από τις υπηρεσίες που παρέχει απευθείας στον τελικό χρήστη διατίθεται και σε έκδοση API απευθυνόμενη σε προγραμματιστές. Διατίθεται σε τρεις εκδόσεις, Standard, Prime και Elite, ενώ διατίθεται και δωρεάν έκδοση Limited Trial με τον περιορισμό των 50 κλήσεων ανά ημέρα.

Υπηρεσίες	**Limited Trial**	**Standard**	**Prime**	**Elite**
Τοποθεσίες	✓	✓	✓	✓
Τρέχουσες καιρικές συνθήκες	✓	✓	✓	✓
Ιστορικό συνθήκες 24ων ωρών	✓	✓	✓	✓
Προβλέψεις ανά ημέρα	5 ημερών	5 ημερών	10 ημερών	15 ημερών
Προβλέψεις ανά ώρα	Ανά 12 ώρες	Ανά 12 ώρες	Ανά 72 ώρες	Ανά 120 ώρες
Δείκτες	5 ημερών	5 ημερών	10 ημερών	15 ημερών
Προειδοποιήσεις	✗	5 ημερών	10 ημερών	15 ημερών
Μεταφράσεις	✗	✓	✓	✓
Τροπικά δεδομένα	✗	✗	✓	✓
Ειδοποιήσεις	✗	✗	✓	✓
Εικόνες	✗	✗	✓	✓
Τιμολόγηση	**Δωρεάν έως 50 κλήσεις/ ημέρα 1 κλειδί/ προγραμματιστή**	**25$/ μήνα + 0.12$/ χίλιες κλήσεις, πάνω από 225000**	**250$/ μήνα + 0.15$/ χίλιες κλήσεις, πάνω από 1800000**	**500$/ μήνα + 0.22$/ χίλιες κλήσεις, πάνω από 2400000**

Πίνακας 19: Χαρακτηριστικά εκδόσεων του AccuWeather [38]

Μετά από την εγγραφή στην πλατφόρμα παρέχεται ένα μοναδικό κλειδί Api key, μέσω του οποίου γίνεται η κλήση του API προσδιορίζοντας τις κατάλληλες παραμέτρους. Το API καλείτε μέσω ενός HTTP GET Request το οποίο επιστρέφει ένα JSON με όλα τα δεδομένα.

Για την λήψη των τρεχουσών καιρικών συνθηκών απαιτείτε ο προσδιορισμός της περιοχής με το κωδικό της καθώς και η χρήση του Api key.

```
http://dataservice.accuweather.com/currentconditions/v1/184896?apikey=
API_KEY&details=true
```

Τέλος επιστρέφεται το παρακάτω JSON το οποίο δημιουργεί αντικείμενα που περιλαμβάνουν όλες τις απαραίτητες πληροφορίες των τρεχουσών καιρικών συνθηκών.

- [] JSON
  - {} 0
    - LocalObservationDateTime : "2021-05-30T19:31:00+03:00"
    - EpochTime : 1622392260
    - WeatherText : "Rain"
    - WeatherIcon : 18
    - HasPrecipitation : true
    - PrecipitationType : "Rain"
    - IsDayTime : true
    - {} Temperature
    - {} RealFeelTemperature
    - {} RealFeelTemperatureShade
    - RelativeHumidity : 93
    - IndoorRelativeHumidity : 61
    - {} DewPoint
    - {} Wind
    - {} WindGust
    - UVIndex : 0
    - UVIndexText : "Low"
    - {} Visibility
    - ObstructionsToVisibility : "R-"
    - CloudCover : 77
    - {} Ceiling
    - {} Pressure
    - {} PressureTendency
    - {} Past24HourTemperatureDeparture
    - {} ApparentTemperature
    - {} WindChillTemperature
    - {} WetBulbTemperature
    - {} Precip1hr
    - {} PrecipitationSummary
    - {} TemperatureSummary
    - MobileLink : "http://m.accuweather.com/en/gr/kozani/184896/current-weather/184896?lang=en-us"
    - Link : "http://www.accuweather.com/en/gr/kozani/184896/current-weather/184896?lang=en-us"

Όσον αφορά τη λήψη μετεωρολογικών προγνώσεων απαιτείτε και πάλι ο προσδιορισμός της περιοχής με το κωδικό της καθώς και η χρήση του Api key. Ακολουθεί παράδειγμα κλήσης του API για λήψη πρόγνωσης για τις επόμενες 5 ημέρες.

`http://dataservice.accuweather.com/forecasts/v1/daily/5day/`**`184896`**`?apikey=`**`API_KEY`**`&details=true&metric=true`

Επιστρέφεται και πάλι ένα JSON αντικείμενο το οποίο περιέχει την πρόγνωση των επόμενων 5 ημερών. Πιο συγκεκριμένα, δημιουργείται αντικείμενο `Headline` το οποίο περιέχει γενικότερες πληροφορίες που αφορούν τη συγκεκριμένη περιοχή καθώς και σύντομη μετεωρολογική περιγραφή.

- JSON
  - Headline
    - EffectiveDate : "2021-05-30T20:00:00+03:00"
    - EffectiveEpochDate : 1622394000
    - Severity : 3
    - Text : "Showers and thunderstorms around Sunday evening through Monday evening"
    - Category : "thunderstorm"
    - EndDate : "2021-06-01T02:00:00+03:00"
    - EndEpochDate : 1622502000
    - MobileLink : "http://m.accuweather.com/en/gr/kozani/184896/extended-weather-forecast/184896?unit=c&lang=en-us"
    - Link : "http://www.accuweather.com/en/gr/kozani/184896/daily-weather-forecast/184896?unit=c&lang=en-us"
  - DailyForecasts
    - 0
    - 1
    - 2
    - 3
    - 4

Όσον αφορά τον πίνακα `DailyForecasts`, δημιουργεί πέντε αντικείμενα από το 0 έως το 4 κάθε ένα από τα οποία αφορά την ημερήσια πρόγνωση για τις επόμενες πέντε ημέρες. Το κάθε αντικείμενο δομείτε από άλλα αντικείμενα για την ανάλυση των δεδομένων της κάθε μετεωρολογικής τιμής.

- 0
  - Date : "2021-05-30T07:00:00+03:00"
  - EpochDate : 1622347200
  - Sun
  - Moon
  - Temperature
  - RealFeelTemperature
  - RealFeelTemperatureShade
  - HoursOfSun : 4.1
  - DegreeDaySummary
  - AirAndPollen
  - Day
  - Night

- Sources
  - MobileLink : "http://m.accuweather.com/en/gr/kozani/184896/daily-weather-forecast/184896?day=1&unit=c&lang=en-us"
  - Link : "http://www.accuweather.com/en/gr/kozani/184896/daily-weather-forecast/184896?day=1&unit=c&lang=en-us"

5.1.4 Weather Underground

Εικόνα 94: Λογότυπο Weather Underground

Ακριβώς όπως και οι πλατφόρμες που έχουν ήδη αναφερθεί έτσι και το Weather Underground (wunerground) αποτελεί μια μετεωρολογική πλατφόρμα παρέχοντας στους χρήστες δεδομένα καιρικών συνθηκών που επικρατούν σε διάφορες περιοχές. Μια και τα δεδομένα καιρικών μετρήσεων βασίζονται κυρίως στην συνεισφορά της κοινότητας, η πλατφόρμα ανταμείβει τους χρήστες που μοιράζονται μαζί της μετεωρολογικά δεδομένα από προσωπικούς σταθμούς μέσω της API έκδοσης. Η API έκδοση που προσφέρεται εξ ’ολοκλήρου δωρεάν στους παραπάνω χρήστες παρέχει δεδομένα ιστορικών, τρεχουσών και προγνωστικών καιρικών συνθηκών. Τα δεδομένα που παρέχει το API επιστρέφονται σε μορφή JSON από προεπιλογή μετά από ένα HTTP GET Request.

Για την λήψη της τρέχουσας κατάστασης καιρικών συνθηκών καλείται το API με προσδιορισμό του μετεωρολογικού σταθμού, της μονάδας μέτρησης και του μοναδικού κλειδιού APIKey.

`https://api.weather.com/v2/pws/observations/current?`**`stationId=IKOZAN2`**`&format=json&`**`units=m`**`&numericPrecision=decimal&apiKey=`**`APIKey`**

Το API επιστρέφει κατάλληλο JSON το οποίο περιέχει αντικείμενο με τις τιμές προσδιορισμού του τόπου και χρόνου των μετρήσεων καθώς και αντικείμενο με όλες τις καιρικές μετρήσεις.

- JSON
  - observations
    - 0
      - stationID : "IKOZAN2"
      - obsTimeUtc : "2021-05-30T19:54:42Z"
      - obsTimeLocal : "2021-05-30 22:54:42"
      - neighborhood : "Kozani"
      - softwareType : null

- country : "GR"
- solarRadiation : null
- lon : 21.787136
- realtimeFrequency : null
- epoch : 1622404482
- lat : 40.302437
- uv : null
- winddir : 315
- humidity : 93
- qcStatus : -1
- {} metric
  - temp : 12.6
  - heatIndex : 12.6
  - dewpt : 11.5
  - windChill : 12.6
  - windSpeed : 9.3
  - windGust : 9.7
  - pressure : 1009.82
  - precipRate : 3.81
  - precipTotal : 9.14
  - elev : 719.9

Για την λήψη της πρόγνωσης των καιρικών φαινομένων για τις επόμενες πέντε ημέρες το API καλείται με τον παρακάτω τρόπο.

`https://api.weather.com/v3/wx/forecast/daily/5day?geocode=`**`40.3,21.78`**`&format=json&`**`units=m`**`&language=en-US&apiKey=` **`APIKey`**

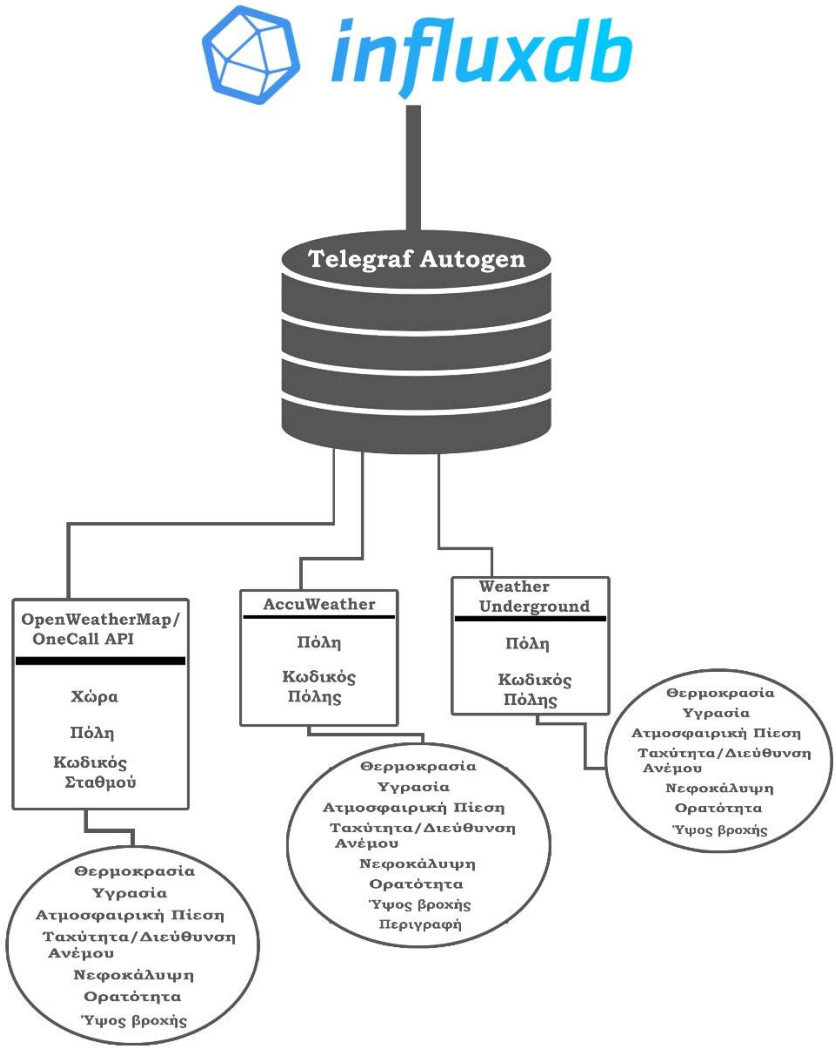
Το API επιστρέφει JSON το οποίο περιέχει δεδομένα πρόγνωσης καιρού για τις επόμενες πέντε ημέρες στη συγκεκριμένη περιοχή.

- {} JSON
  - [] calendarDayTemperatureMax
  - [] calendarDayTemperatureMin
  - [] dayOfWeek
  - [] expirationTimeUtc
  - [] moonPhase
  - [] moonPhaseCode
  - [] moonPhaseDay
  - [] moonriseTimeLocal
  - [] moonriseTimeUtc
  - [] moonsetTimeLocal
  - [] moonsetTimeUtc
  - [] narrative
  - [] qpf
  - [] qpfSnow

- [ ] sunriseTimeLocal
- [ ] sunriseTimeUtc
- [ ] sunsetTimeLocal
- [ ] sunsetTimeUtc
- [ ] temperatureMax
- [ ] temperatureMin
- [ ] validTimeLocal
- [ ] validTimeUtc
- [ ] daypart

5.2 Συλλογή δεδομένων API's

Για τις ανάγκες αποθήκευσης, προώθησης και οπτικοποίησης των δεδομένων από τις πλατφόρμες που εξηγήθηκαν προηγουμένως έγινε και πάλι χρήση της InfluxDB τόσο στον τοπικό εξυπηρετητή όσο και στο InfluxDB Cloud, του Telegraf και του Grafana.

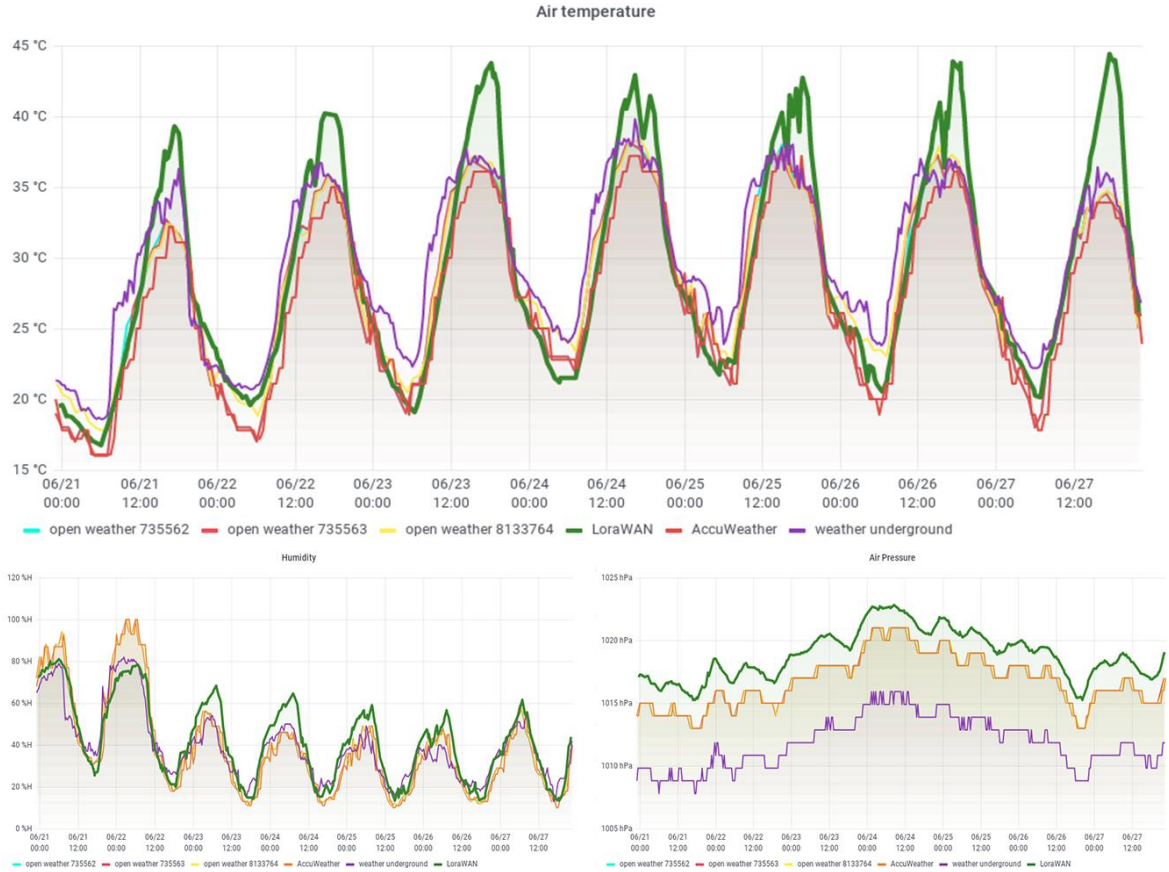


Εικόνα 95: Σχηματικό διάγραμμα βάσης δεδομένων

Εικόνα 96: Ροή δεδομένων API's

5.3 Συσχέτιση

Πραγματοποιήθηκε συσχέτιση των δεδομένων ανοιχτών πηγών που συλλέγονται στη βάση δεδομένων με αυτά που καταγράφει ο αισθητήριος κόμβος με σκοπό την αξιολόγησή τους και τελικό στόχο την λήψη συμπερασμάτων. Πιο συγκεκριμένα, τα δεδομένα τα οποία υποβλήθηκαν σε σύγκριση αποτέλεσαν οι μετρήσεις ατμοσφαιρικής θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης, ενώ πηγές ανοιχτών δεδομένων αποτέλεσε το OpenWeatherMap μέσω των σταθμών Kozani 735562, 735563, 8133764, το AccuWeather και το Weather Underground. Σκοπό της συσχέτισης αποτέλεσε η ανάγκη άντλησης ενός συμπεράσματος σχετικά με την ταύτιση των δεδομένων του αγρομετεωρολογικού σταθμού με αυτά των API's. Για την επεξεργασία και την οπτικοποίηση των παραπάνω δεδομένων χρονοσειρών έγινε χρήση του λογισμικού Grafana. Ακολουθούν κοινά διαγράμματα των δεδομένων στην ίδια χρονική περίοδο σε διάστημα επτά ημερών.



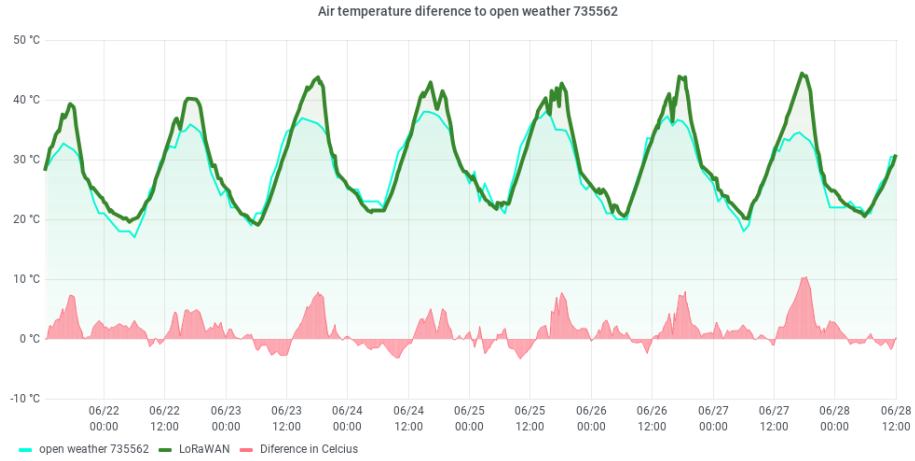
Εικόνα 97: Κοινά διαγράμματα αποτύπωση δεδομένων θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης του αισθητήριου κόμβου με αυτά των API's

Τα δεδομένα του αισθητήριου κόμβου παρατηρήθηκε να σημειώνουν μεγάλες ομοιότητες με αυτά των API's τόσο ρυθμό μεταβολής τους όσο και στο μέτρο αυτής. Αξίζει να σημειωθεί σε πρώτη φάση πως παρατηρήθηκε η αυξητική τάση της θερμοκρασίας του αισθητήριου κόμβου σε σχέση με τα API's από 12:00 έως 20:00 και ιδιαίτερα της ημέρες με ηλιοφάνεια. Μια πιθανή εξήγηση αποτελεί ένα σύνηθες πρόβλημα το οποίο αφορά την εγκατάσταση του αισθητήρα, μια και πολλές φορές η εγκατάσταση σε μη σκιερά μέρη ή η χρήση υλικών που δεν ανακλούν πλήρως την ηλιακή ακτινοβολία οδηγούν σε λάθος μέτρηση. Σε καμία περίπτωση δεν προβλέπεται η απόλυτη ταύτιση των μετρήσεων μια και παράγοντες τοπικότητας επηρεάζουν σημαντικά τις μετρήσεις μιας περιοχής.

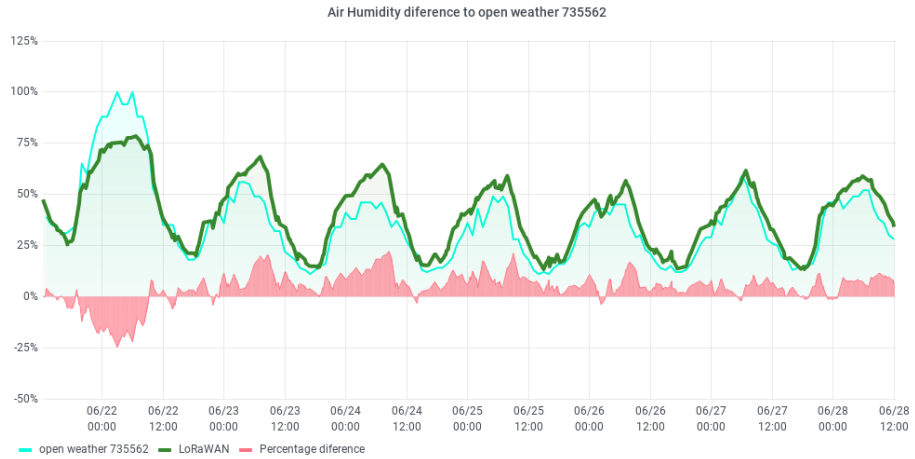
5.3.1 OpenWeatherMap

5.3.1.1 Σταθμός 735562

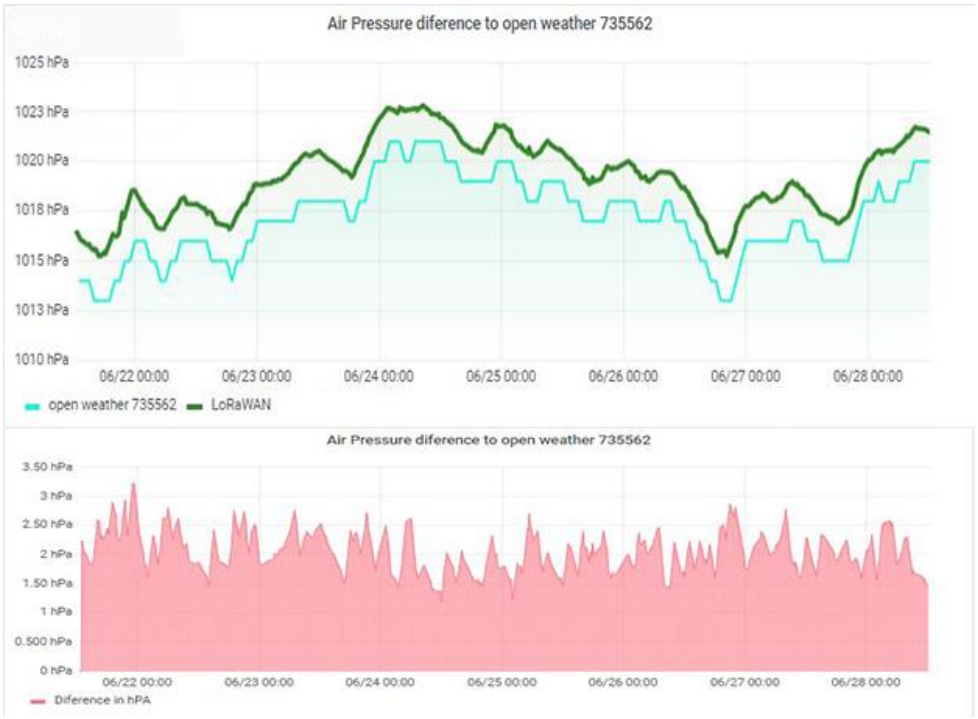
Μετά από σύγκριση των δεδομένων του αισθητήριου κόμβου με τον σταθμό 735562 που παρέχεται από το OpenWeatherMap υπολογίστηκε η μέση διαφορά των μετρήσεων της θερμοκρασίας σε 2.09 °C , της υγρασίας σε 7.18% ενώ της ταχύτητας της ατμοσφαιρικής πίεσης σε 2.05 hPa Ακολουθούν κοινά διαγράμματα οπτικοποίησης των μετρήσεων καθώς και της διαφοράς τους σε σχέση με την πάροδο του χρόνου.



Εικόνα 98: Διαφορά τιμών θερμοκρασίας συγκριτικά με τον σταθμό 735562



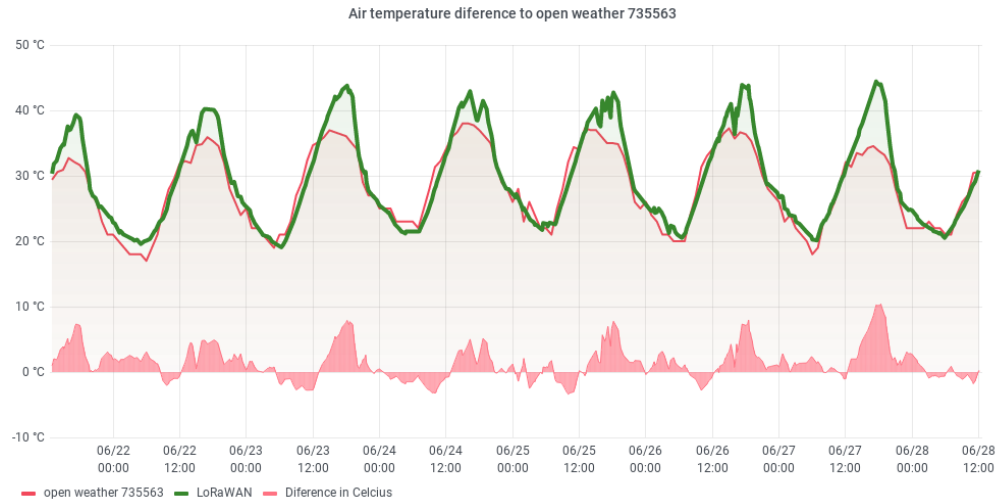
Εικόνα 99: Διαφορά τιμών υγρασίας συγκριτικά με τον σταθμό 735562



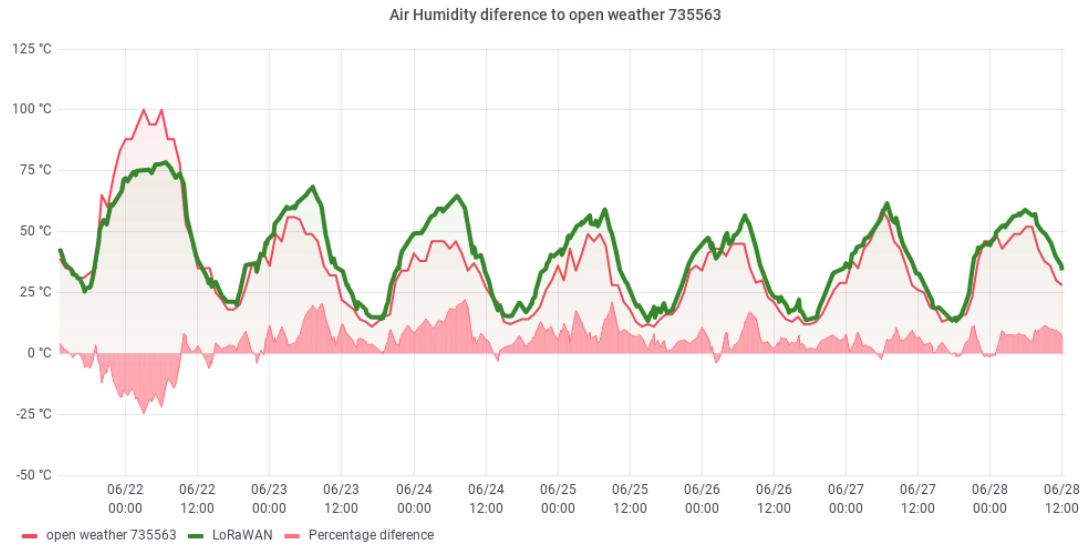
Εικόνα 100: Διαφορά τιμών ατμοσφαιρικής πίεσης συγκριτικά με τον σταθμό 735562

5.3.1.2 Σταθμός 735563

Μετά από σύγκριση των δεδομένων του αισθητήριου κόμβου με τον σταθμό 735563 που παρέχεται από το OpenWeatherMap υπολογίστηκε η μέση διαφορά των μετρήσεων της θερμοκρασίας σε 2.14 °C , της υγρασίας σε 7.18% ενώ της ταχύτητας της ατμοσφαιρικής πίεσης σε 2.05 hPa Ακολουθούν κοινά διαγράμματα οπτικοποίησης των μετρήσεων καθώς και της διαφοράς τους σε σχέση με την πάροδο του χρόνου.



Εικόνα 101: Διαφορά τιμών θερμοκρασίας συγκριτικά με τον σταθμό 735563

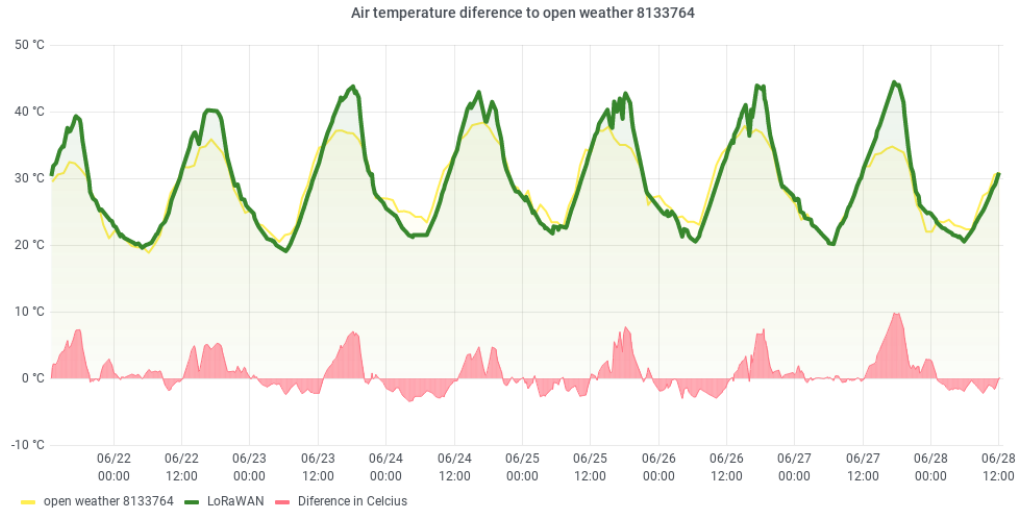


Εικόνα 102: Διαφορά τιμών υγρασίας συγκριτικά με τον σταθμό 735563

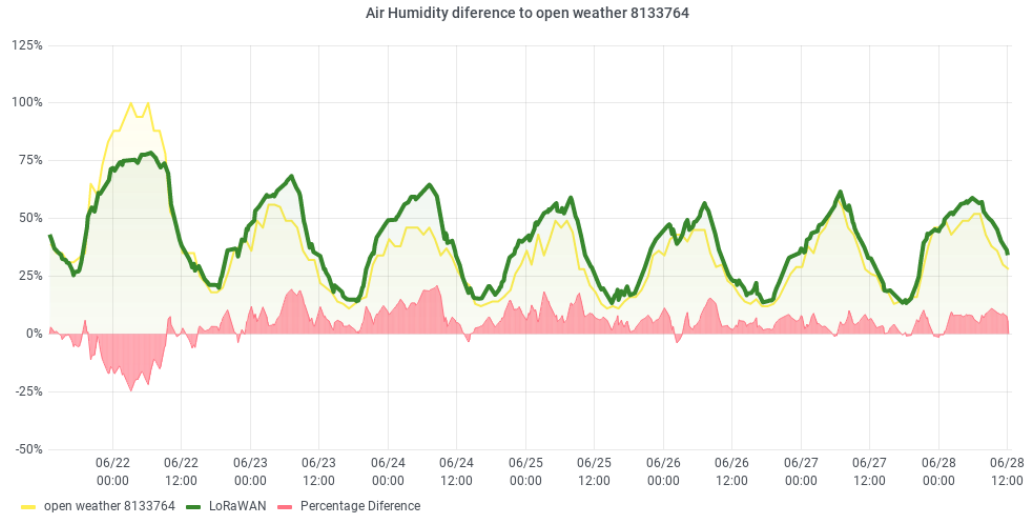
Εικόνα 103: Διαφορά τιμών ατμοσφαιρικής πίεσης συγκριτικά με τον σταθμό 735563

5.3.1.3 Σταθμός 8133764

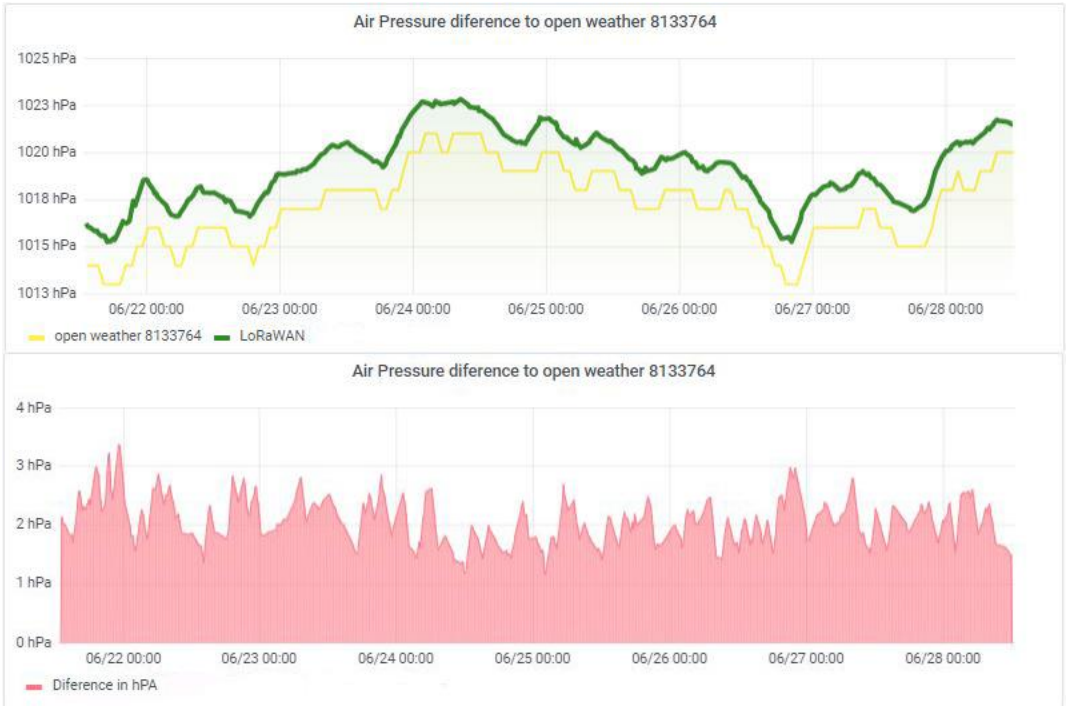
Μετά από σύγκριση των δεδομένων του αισθητήριου κόμβου με τον σταθμό 8133764 που παρέχεται από το OpenWeatherMap υπολογίστηκε η μέση διαφορά των μετρήσεων της θερμοκρασίας σε 2.02 °C , της υγρασίας σε 7.10% ενώ της ταχύτητας της ατμοσφαιρικής πίεσης σε 2.06 hPa Ακολουθούν κοινά διαγράμματα οπτικοποίησης των μετρήσεων καθώς και της διαφοράς τους σε σχέση με την πάροδο του χρόνου.



Εικόνα 104: Διαφορά τιμών θερμοκρασίας συγκριτικά με τον σταθμό 8133764



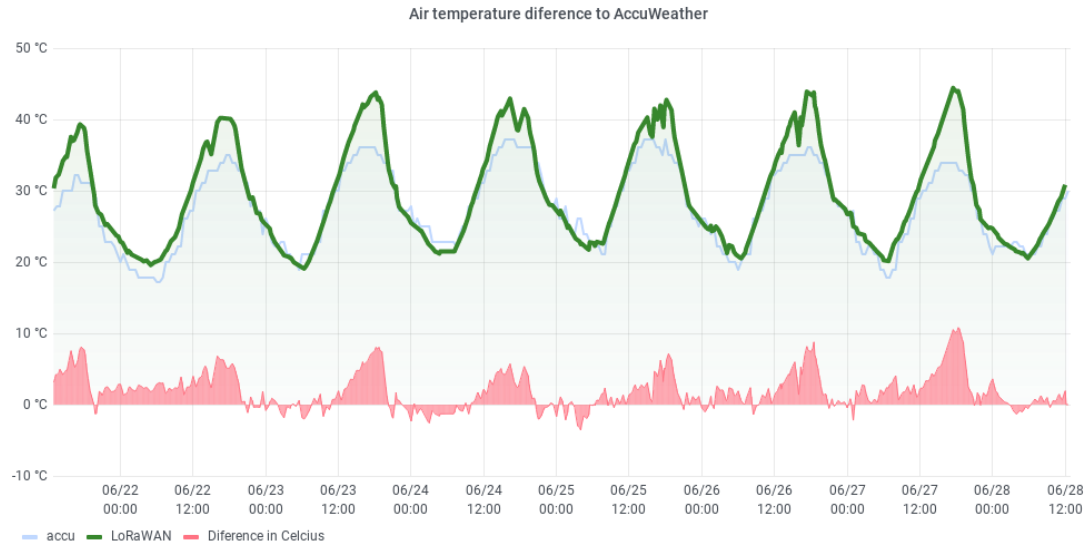
Εικόνα 105: Διαφορά τιμών υγρασίας συγκριτικά με τον σταθμό 8133764



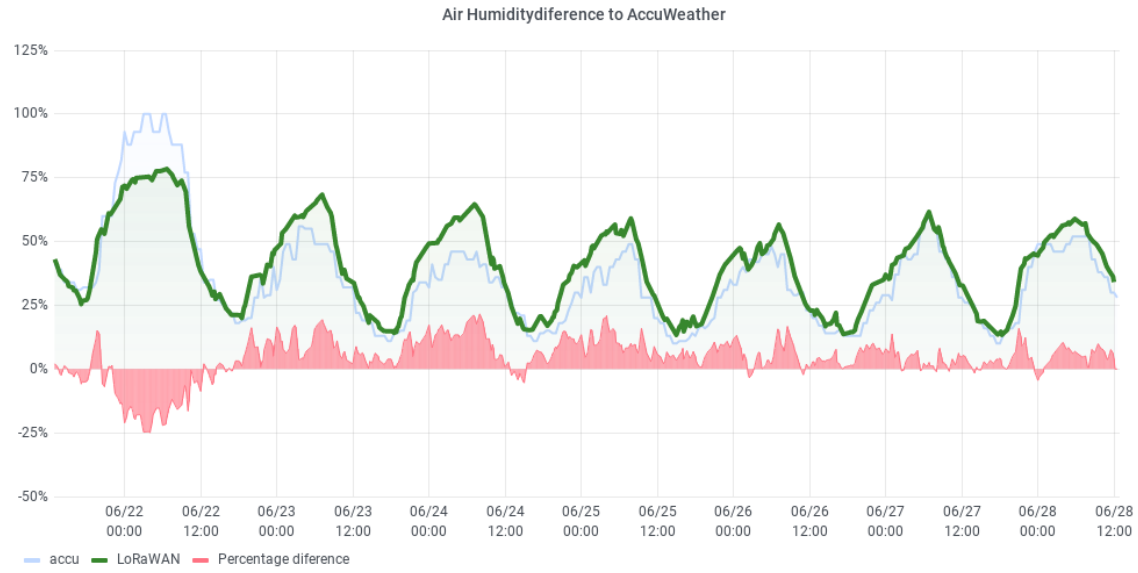
Εικόνα 106: Διαφορά τιμών ατμοσφαιρικής πίεσης συγκριτικά με τον σταθμό 8133764

5.3.2 AccuWeather

Σε σύγκριση με τα δεδομένα του AccuWeather υπολογίστηκε η μέση διαφορά των μετρήσεων της θερμοκρασίας σε 2.30 °C , της υγρασίας σε 7.33% ενώ της ταχύτητας της ατμοσφαιρικής πίεσης σε 2.04 hPa Ακολουθούν κοινά διαγράμματα οπτικοποίησης των μετρήσεων καθώς και της διαφοράς τους σε σχέση με την πάροδο του χρόνου.



Εικόνα 107: Διαφορά τιμών θερμοκρασίας συγκριτικά με το AccuWeather

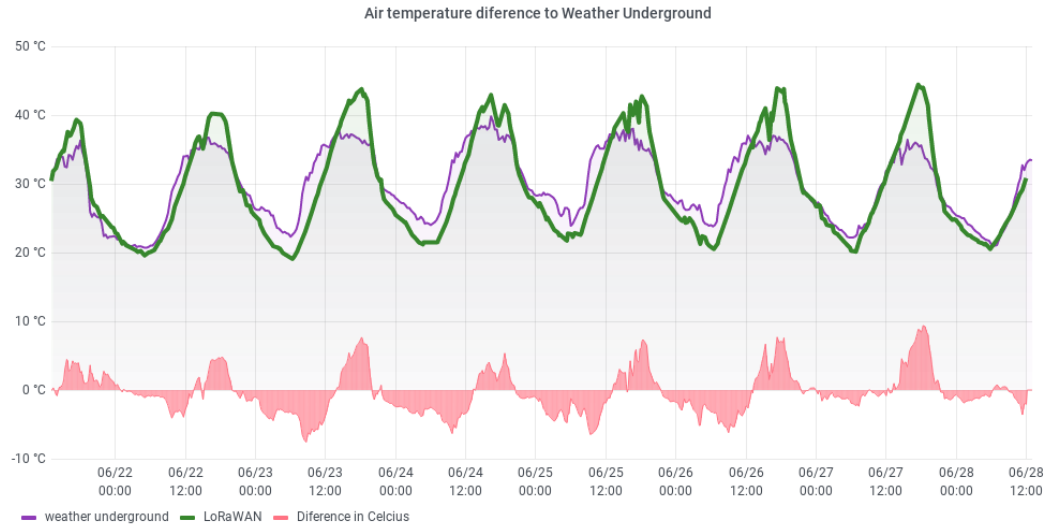


Εικόνα 108: Διαφορά τιμών υγρασίας συγκριτικά με το AccuWeather

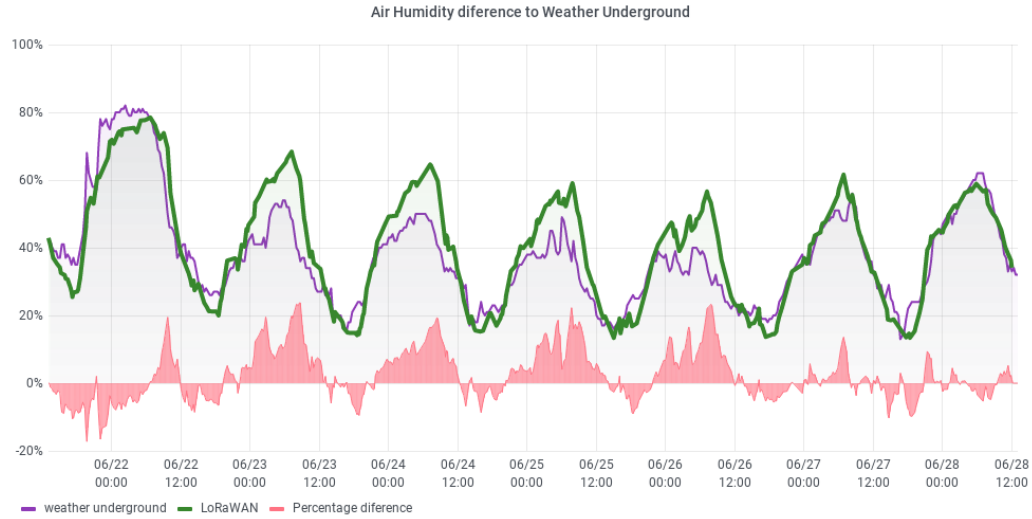
Εικόνα 109: Διαφορά τιμών ατμοσφαιρικής πίεσης συγκριτικά με το AccuWeather

5.3.3 Weather Underground

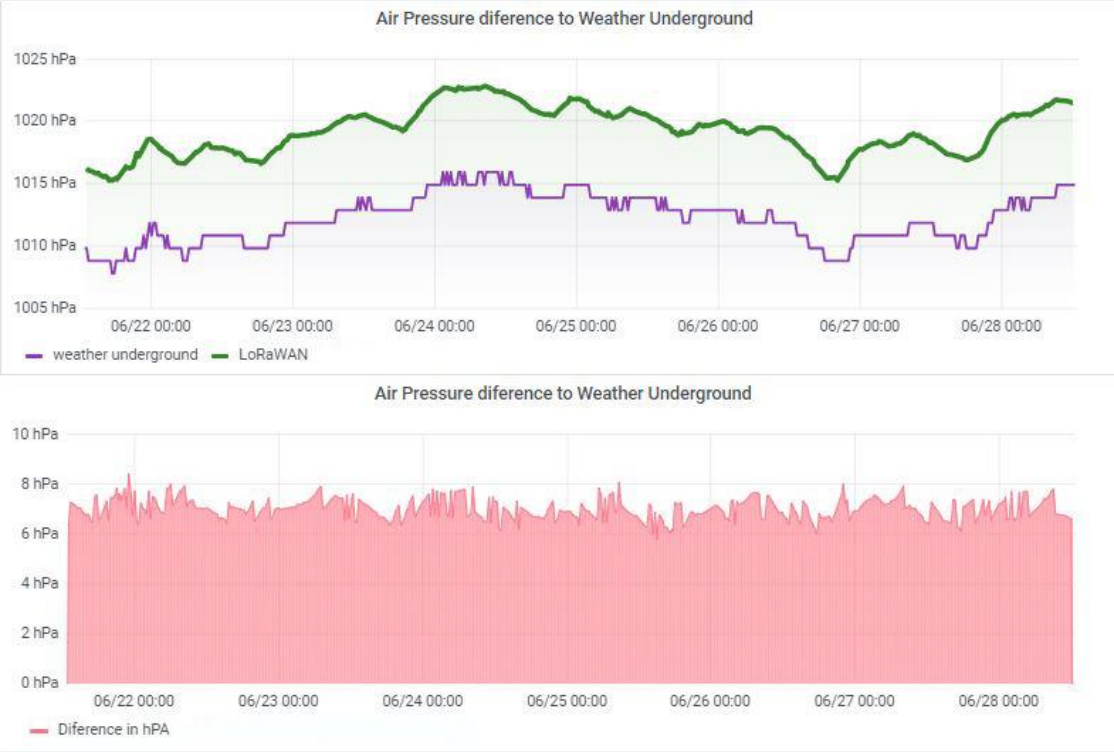
Όσον αφορά τα δεδομένα του Weather Underground υπολογίστηκε η μέση διαφορά των μετρήσεων της θερμοκρασίας σε 2.52 °C , της υγρασίας σε 5.88% ενώ της ταχύτητας της ατμοσφαιρικής πίεσης σε 6.97 hPa Ακολουθούν κοινά διαγράμματα οπτικοποίησης των μετρήσεων καθώς και της διαφοράς τους σε σχέση με την πάροδο του χρόνου..



Εικόνα 110: Διαφορά τιμών θερμοκρασίας συγκριτικά με το Weather Underground



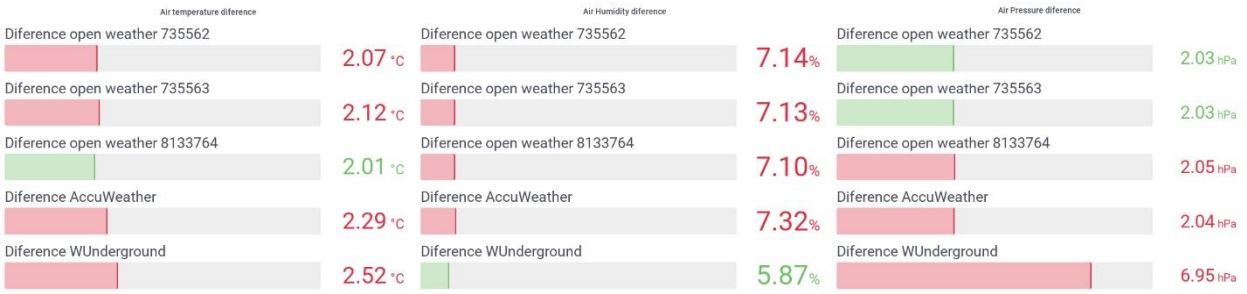
Εικόνα 111: Διαφορά τιμών υγρασίας συγκριτικά με το Weather Underground



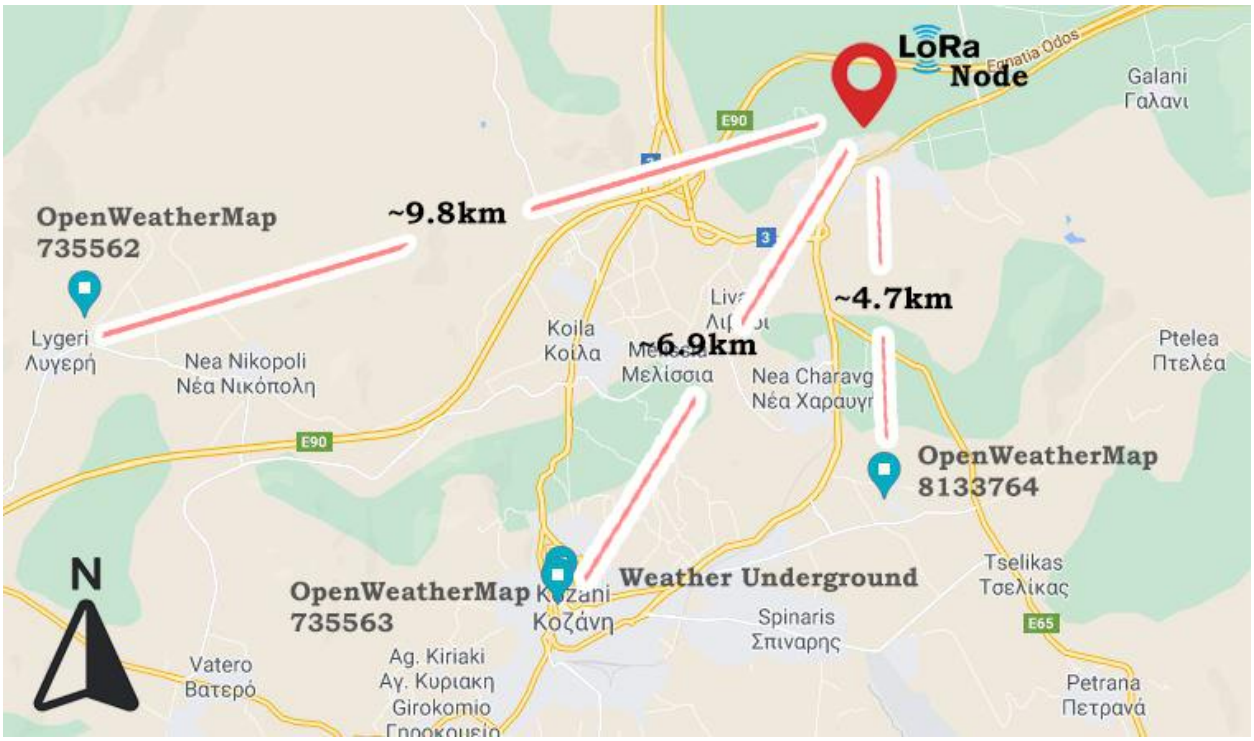
Εικόνα 112: Διαφορά τιμών ατμοσφαιρικής πίεσης συγκριτικά με το Weather Underground

5.4 Σύνοψη

Συνοψίζοντας, μετά από ανάλυση όλων των παραπάνω δεδομένων σημειώθηκαν αρκετές παρατηρήσεις σχετικά με την συσχέτιση των δεδομένων του αισθητήριου κόμβου με αυτά πηγών ανοιχτών δεδομένων. Παρατηρήθηκαν συμπεράσματα εποχικότητας, όπως αυτό της αυξητικής τάσης της θερμοκρασίας σε συγκεκριμένες ώρες, αλλά και μεγάλες ομοιότητες χωρίς σημαντικές διαφορές. Ο αισθητήριος κόμβος παρουσίασε τις μικρότερες διαφορές με τον σταθμό 8133764 του OpenWeatherMap για τις τιμές ατμοσφαιρικής θερμοκρασίας, με τα δεδομένα του Weather Underground για τις τιμές της ατμοσφαιρικής υγρασίας , ενώ όσον αφορά την ατμοσφαιρική πίεση η μικρότερες αποκλίσεις παρατηρήθηκαν με του σταθμούς 735562 και 735563 του OpenWeatherMap.



Εικόνα 113: Διαφορά των τιμών του αισθητήριου κόμβου με αυτές των API's



Εικόνα 114: Γεωγραφική τοποθέτηση αισθητήριου κόμβου και πηγών API's

Επίλογος

Συνοψίζοντας, με το πέρας της διπλωματικής εργασίας επιτεύχθηκαν οι αρχικοί στόχοι που είχαν τεθεί, αντιμετωπίστηκαν διάφορα προβλήματα και τελικά υλοποιήθηκε μια ολοκληρωμένη λύση παρακολούθησης γεωργικών εκτάσεων. Έγινε κατανόηση και χρήση των εννοιών της έξυπνης γεωργίας καθώς και του διαδικτύου των πραγμάτων (IoT). Όσον αφορά την υλοποίηση του αισθητήριου κόμβου, έγινε προσεκτική μελέτη ώστε να εξαλειφθούν πιθανές αδυναμίες, ενώ πολλές φορές και ιδιαίτερα στην διαδικασία της εφαρμογής σε πραγματικές συνθήκες, αντιμετωπίστηκαν διάφορα προβλήματα. Η χρήση της τεχνολογίας LoRa κρίνεται πολύ σημαντική για χρήση στην έξυπνη γεωργία καθώς δίνει λύσεις στα προβλήματα κάλυψης και κατανάλωσης ενέργειας, τα οποία οι συμβατικές ασύρματες τεχνολογίες δεν μπορούν να αντιμετωπίσουν. Η χρήση της πλατφόρμας The Things Network ήταν επίσης πολύ σημαντική η οποία παρέχει δυνατότητες για τη σωστή διαχείριση των δεδομένων που αντλούνται μέσω του LoRaWAN. Η χρήση της βάσης δεδομένων InfluxDB, του λογισμικού οπτικοποίησης Grafana, του εργαλείου Telegraf και της υπηρεσίας νέφους Telegram αποδείχθηκε αποτελεσματική. Η δημιουργία ενός micro-server βασισμένο σε μια συσκευή χαμηλού κόστους Raspberry Pi και η ενοποίηση όλων των παραπάνω σε αυτό αποδείχθηκε επίσης ιδιαίτερα αξιόπιστη. Επίσης, ιδιαίτερα σημαντική ήταν και η διαδικασία συσχέτισης των δεδομένων του πρωτότυπου αισθητήριου κόμβου LoRaWAN με αυτά μετεωρολογικών πηγών μια και αντλήθηκαν συμπεράσματα αξιοπιστίας .

Η συγκεκριμένη λύση παρακολούθησης γεωργικών εκτάσεων μπορεί άμεσα να χρησιμοποιηθεί για τις ανάγκες ενός γεωργού καθώς και να προσαρμοστεί σε τυχών ιδιαιτερότητες, όπως η χρήση διαφορετικών αισθητηρίων και η πιο τακτική λήψη δεδομένων. Είναι πολύ σημαντικό να αναφερθεί ότι για τη χρήση της συγκεκριμένης λύσης δεν απαιτούνται μηνιαία κόστη, όπως χρήση τηλεφωνικών δικτύων (κάρτες SIM) ή επί πληρωμή πλατφόρμες, καθώς η λύση βασίζεται εξ ολοκλήρου στη χρήση του δικτύου LoRa και σε λογισμικά και υπηρεσίες ανοικτού κώδικα.

Προβλήματα που αντιμετωπίστηκαν

Στη διάρκεια της τεχνικής υλοποίησης ήταν αρκετές οι φορές που έπρεπε να ξεπεραστούν εμπόδια και να βρεθούν λύσεις. Μετά την επιλογή των εξαρτημάτων έπρεπε να βρεθεί τρόπος να κατασκευαστεί ο αισθητήριος κόμβος με τρόπο τέτοιο ώστε να καταλαμβάνει όσο το δυνατόν λιγότερο χώρο και να είναι προστατευμένος από τις καιρικές συνθήκες. Τη λύση έδωσε η τρισδιάστατη εκτύπωση μέσω της δυνατότητας της σχεδίασης και της παραγωγής εξαρτημάτων στοχεύοντας στις συγκεκριμένες ανάγκες μιας κατασκευής.

Με την επιλογή του Raspberry Pi ως εξυπηρετητή για την εκτέλεση των διεργασιών αποθήκευσης και οπτικοποίησης των δεδομένων παρουσιάστηκε μια σημαντική πρόκληση η οποία αφορούσε την διαδικασία rendering του Grafana, η συγκεκριμένη διαδικασία ήταν σημαντική και είχε τεθεί ως στόχος από την αρχή για την αποστολή γραφημάτων μέσω του Telegram. Πιο συγκεκριμένα, το λογισμικό Grafana θέτει ως προϋπόθεση για την εγκατάσταση του rendering Plugin την εκτέλεσή του λογισμικά σε Linux (x64) ή Windows (x64) ή Mac OS X (x64) αλλά όχι σε διανομές Debian όπως αυτή του Raspberry OS που εκτελείται στον εξυπηρετητή micro-server. Η συγκεκριμένη πρόκληση αντιμετωπίστηκε επιτυχώς μετά από μια σειρά δοκιμών και το Raspberry Pi αποδείχθηκε πως διαθέτει όλη την απαραίτητη υπολογιστική ισχύ για να εκτελέσει διεργασίες rendering μέσω του Grafana.

Πολύ σημαντική ήταν η εφαρμογή της λύσης σε πραγματικές συνθήκες για την παρακολούθηση γεωργικών εκτάσεων καθώς προέκυψαν και αντιμετωπίστηκαν προβλήματα που δεν εμφανίστηκαν σε εργαστηριακές συνθήκες. Αρχικά προέκυψε η ανάγκη αποστολής σταθερού payload μηνύματος από τον αισθητήριο κόμβο, η οποία δεν είχε παρατηρηθεί στις μικρές μεταβολές των μετρήσεων του εσωτερικού χώρου εργασίας. Όσον αφορά την αξιοπιστία, στις πραγματικές συνθήκες αποδείχθηκε πως ορισμένοι αισθητήρες αντιμετώπιζαν προβλήματα σε βάθος χρόνου και αντικαταστάθηκαν με διαφορετικούς.

Μελλοντικές επεκτάσεις

Παρά τις δοκιμές και τις διορθώσεις που προέκυψαν σε όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας είναι σίγουρο πως πάντοτε υπάρχει χώρος για βελτίωση και μεγαλύτερη αξιοποίηση μιας εφαρμογής προσαρμόζοντας στις ανάγκες των εκάστοτε συνθηκών. Ένας από τους αρχικούς στόχους ήταν η υλοποίηση ενός συστήματος εύκολα επεκτάσιμου, αυτό επιτεύχθηκε μια και όπως αποδεικνύεται από το τέταρτο κεφάλαιο της «Εφαρμογής» προσαρμόστηκαν με μεγάλη ευκολία νέοι αισθητήρες των οποίων η εγκατάσταση δεν προέκυπτε από το αρχικό πλάνο. Η συγκεκριμένη λύση που όσον αφορά το «υλικό», αποτελείτε από έναν αισθητήριο κόμβο και έναν εξυπηρετητή, μπορεί να παρέχει πολύ σημαντικές πληροφορίες μεγάλης σημασίας στον τελικό χρήστη για τις συνθήκες αποκλειστικά σε ένα σημείο μιας γεωργικής έκτασης. Παρόλα αυτά θα ήταν εξαιρετικά σημαντικό, και ιδιαίτερα σε μεγάλες εκτάσεις, να υλοποιηθεί ένα δίκτυο αισθητήριων κόμβων.

Περεταίρω αξιοποίηση του αισθητήριου κόμβου μπορεί να γίνει με την ενσωμάτωση μηχανισμών αυτοματισμών, όπως για παράδειγμα μηχανισμούς διαχείρισης ανακυκλώσιμου νερού. Ο μικροελεγκτής SODAQ ExpLoRer διαθέτει τόσο εξόδους όσο και εισόδους που χρησιμοποιούνται για την συλλογή δεδομένων από τους αισθητήρες. Μπορεί να προγραμματιστεί κατάλληλα για τον απομακρυσμένο έλεγχο μηχανισμών όπως αυτοί του ελέγχου ηλεκτρικών βαλβίδων ροής νερού.

Αν και η αλληλεπίδραση μέσω του Telegram είναι αποτελεσματική, άμεση και χωρίς κόστος τουλάχιστον για έναν αισθητήριο κόμβο, θα ήταν ιδανικό να αναπτυχθεί κατάλληλη web πλατφόρμα για την ενσωμάτωση όλων των πληροφοριών. Αντίστοιχα, στην περίπτωση χρήσης της λύσης από διαφορετικούς παραγωγούς ,όπως για παράδειγμα στη χρήση από έναν αγροτικό συνεταιρισμό, θα ήταν ιδανική η χρήση ενός εξυπηρετητή με περισσότερη υπολογιστική ισχύ.

Όσον αφορά τα δεδομένα των μετρήσεων, αυτή τη στιγμή η αξιοποίησή τους παραμένει στη χρήση για προβολή στατιστικών, αποστολή ειδοποιήσεων και συσχέτιση με άλλες πηγές, μια μελλοντική επέκταση θα αποτελούσε η επιπλέον αξιοποίηση τους για τη χρήση σε μετεωρολογικά μοντέλα.

Βιβλιογραφία

[1] Food and Agriculture Organization (FAO), «The future of food and agriculture: Trends and challenges,» p. http://www.fao.org/3/i6583e/i6583e.pdf, 2017.

[2] Project Augeias, «https://project-augeias.gr/».

[3] Ερευνητικό έργο MARS, «https://project-mars.eu/».

[4] JOHN DEERE, «https://www.deere.com/en/technology-products/precision-ag-technology».

[5] ZDNet, Steve Ranger, «What is the IoT? Everything you need to know about the Internet of Things right now,» 2020.

[6] K. Mekki, E. Bajic, F. Chaxel και F. Meyer, «A comparative study of LPWAN technologies for large-scale IoTdeployment,» *sciencedirect,* December 2017.

[7] M. CHEN, Y. MIAO, Y. HAO και K. HWANG, «Narrow Band Internet of Things,» IEEE, 2017.

[8] K. Mekki, E. Bajic, F. Chaxel και F. Meyer, «Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT,» 2018.

[9] The Things Network, «LoRaWAN, https://www.thethingsnetwork.org/docs/lorawan/classes/index.html».

[10] MQTT, Wikipedia, https://en.wikipedia.org/wiki/MQTT.

[11] MQTT.Org, https://mqtt.org/getting-started/.

[12] An itroduction to time series databases, aiven io, https://aiven.io/blog/an-introduction-to-time-series-databases.

[13] Apache Druid, Wikipedia, https://en.wikipedia.org/wiki/Apache_Druid.

[14] EXtremeDB, Wikipedia, https://en.wikipedia.org/wiki/EXtremeDB.

[15] Kdb+, Wikipedia, https://en.wikipedia.org/wiki/Kdb%2B.

[16] Riak, Wikipedia, https://en.wikipedia.org/wiki/Riak.

[17] RRDtool, Wikipedia, https://en.wikipedia.org/wiki/RRDtool.

[18] InfluxDB, Wikipedia, https://en.wikipedia.org/wiki/InfluxDB.

[19] InfluxDB Cloud , https://www.influxdata.com/products/influxdb-cloud/.

[20] Chronograf, InfluxData, https://www.influxdata.com/time-series-platform/chronograf/.

[21] Telegraf, InfluxData, https://www.influxdata.com/time-series-platform/telegraf/.

[22] Getting started with Grafana, https://grafana.com/docs/grafana/latest/getting-started/.

[23] Telegram bot, https://core.telegram.org/bots.

[24] The Things Network, iot-fpms, https://iot-fpms.fandom.com/wiki/The_Things_Network.

[25] Sodaq ExpLoRer, https://support.sodaq.com/Boards/ExpLoRer/.

[26] BME280 Environmental Sensor User Manual, Waveshare, https://www.waveshare.com/w/upload/7/75/BME280_Environmental_Sensor_User_Manual_EN.pdf.

[27] Introduction to ds18b20, theengineeringprojects.com, https://www.theengineeringprojects.com/2019/01/introduction-to-ds18b20.html.

[28] A. Kagalkar, «Smart Irrigation System,» *ijert,* May 2017.

[29] Sodaq getting started, «https://support.sodaq.com/getting_started/».

[30] Lorawan examples , «https://support.sodaq.com/Boards/ExpLoRer/Examples/lorawan/».

[31] Environmental Sensor Demo Code, «https://www.waveshare.com/wiki/File:BME280-Environmental-Sensor-Demo-Code.7z».

[32] Network Architecture, The Things Network, «https://www.thethingsnetwork.org/docs/network/architecture.html».

[33] Pyload format, «https://www.thethingsnetwork.org/forum/t/payload-formats-howto/3441».

[34] Rendering Grafana, Openhub, «https://community.openhab.org/t/tutorial-grafana-rendering-on-raspberry-pi/71777/58».