ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ : ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Thesis

ΥΛΟΠΟΙΗΣΗ ΔΙΚΤΥΟΥ RS485 ΚΑΙ ΔΙΑΣΥΝΔΕΣΗ ΤΟΥ ΜΕ ΔΙΚΤΥΟ LORAWAN ΜΕΣΩ PROTOCOL BUFFER

Φοιτητής: Άγγελος Βογιατζής

Επιβλέπων: Αν. Καθηγητής Παναγιωτάκης Σπυρίδων

ΗΡΑΚΛΕΙΟ 2024

Ευχαριστίες

Θα ήθελα να εκφράσω την βαθύτατη ευγνωμοσύνη μου στους γονείς μου για την αδιάλειπτη στήριξη και ενθάρρυνσή τους καθ' όλη τη διάρκεια των πανεπιστημιακών μου χρόνων. Η αγάπη, η υπομονή και η πίστη τους στις δυνατότητές μου ήταν μια συνεχής πηγή δύναμης και έμπνευσης.

Επίσης, νιώθω την ανάγκη να εκφράσω τις ειλικρινείς μου ευχαριστίες στον καθηγητή μου, Δρ. Σπυρίδων Παναγιωτάκη, ο οποίος υπήρξε ο εισηγητής της πτυχιακής μου εργασίας. Η διορατική του καθοδήγηση, οι εποικοδομητικές παρατηρήσεις του και η αφοσίωσή του υπήρξαν ανεκτίμητα για την επιτυχή ολοκλήρωση αυτής της έρευνας.

Σας ευχαριστώ θερμά για την καθοδήγηση και την υποστήριξη σας.

Περίληψη

Σκοπός της εργασίας ήταν η έρευνα και η αναπτυξη ενος σχετικα ευκολου τροπου για την αυτοματοποίηση ενός θερμοκηπίου με τη δυνατότητα για απομακρυσμένο έλεγχο ενεργοποιητών και την συγκέντρωση δεδομένων με την χρήση του δικτύου LoRaWAN αντι ενός κινητού δικτύου οπως 4g/5g ή σταθερής γραμμής στο θερμοκήπιο. Οι κόμβοι αισθητήρων και ενεργοποιητών χρησιμοποιούν WiFi και MQTT για την τοπική επικοινωνία η οποία γίνεται με ενα edge device, στην περιπτωση μας ενα Raspberry Pi 3, το οποίο παίρνει τον ρόλο του MQTT server αλλα και του LoRaWAN device για επικοινωνια με το service The Things Network.

Οι αισθητήρες και οι ενεργοποιητές χρησιμοποιούν σειριακή επικοινωνία τύπου RS485 και λειτουργούν σε συνδυασμό με μικροελεγκτές τυπου ESP32 για να δημιουργήσουν τους WiFi κόμβους, οι κόμβοι στέλνουν και δέχονται μηνύματα από το edge device, το οποίο επίσης σειριοποιεί δομημένα δεδομένα με την χρήση Protocol Buffers και αποστέλλονται χρησιμοποιώντας το δίκτυο LoRa WAN.

Λέξεις κλειδιά: Έξυπνο θερμοκήπιο, απομακρυσμένος έλεγχος, αισθητήρες, ενεργοποιητές, WiFi, LoRa WAN, Protocol Buffers, The Things Network, ESP, ESP32, RS485, Edge Device, Raspberry Pi, FreeRTOS.

Abstract

The aim of this thesis was to research and develop an easy way to automate a greenhouse with the capability of gathering data and controlling actuators remotely using LoRa WAN instead of mobile networks like 4g/5g or landline connection in the greenhouse. The sensor and actuator nodes use WiFi and MQTT for the local communication with an edge device, in our case a Raspberry Pi 3, which takes the role of the MQTT server and also works at the LoRa WAN device which talks to The Things Network service.

The sensor and actuators use RS485 serial and work together with ESP32 microcontrollers to complete the WiFi nodes, the nodes send and receive messages from the edge device, which also serialises structured data using Protocol Buffers and sends it using LoRa WAN.

Keywords: Smart greenhouse, remote control, sensors, actuators, WiFi, LoRa WAN, Protocol buffers, The Things Network, ESP, ESP32, RS485, Edge Device, Raspberry Pi, FreeRTOS.

Ευχαριστίες ... **2**
Abstract ... **3**
Περίληψη ... **3**
1. Εισαγωγή ... **6**
1.1. Κίνητρα ... 6
1.2. Δομή της εργασίας ... 8
2. State of the art ... **9**
2.1. Internet Of Things γενικά ... 9
2.2. Αισθητήρες ... 10
2.3. Ασύρματα δίκτυα αισθητήρων(WSN) ... 11
2.4. Internet Of Things στην γεωργία ... 13
2.5. Ήδη υπάρχοντα συστήματα της αγοράς ... 15
2.6. RS-485 ... 16
2.7. Protocol Buffers ... 21
2.8. LoRa και LoRaWAN ... 22
2.9. The Things Network ... 39
2.10. MQTT ... 41
2.11. NanoMQ ... 44
3. Σχεδιασμός ... **46**
3.1. Απαιτήσεις ... 46
3.2. Σχεδιασμός του συστήματος ... 47
4. Υλοποίηση ... **51**
4.1. Υλοποίηση hardware ... 51
4.1.1. Σχεδιασμός WiFi κόμβου με αισθητήρα εδάφους / περισταλτική αντλία ... 51
Real Time Clock ... 53
4.1.2. Σχεδιασμός LoRaWAN End Device και MQTT Broker ... 55
4.2. Υλοποίηση Software ... 57
4.2.1. WiFi Κόμβος αισθητήρα ... 58
4.2.1.1. Inclusions και Declarations ... 58
4.2.1.2. Main Function ... 59
4.2.1.3. Main While ... 60
4.2.1.4. MQTT Task ... 61
4.2.1.4.1. sendMessage ... 62
4.2.2. UART Task ... 62
4.2.2.1. GetSoilData ... 63
4.2.3. Real Time Clock Task ... 64
getRTCEpoch ... 64
4.2.4. WiFi Κόμβος Αντλίας ... 64
4.2.5. Raspi LMIC και caching σύστημα ... 70
4.2.6. Protocol Buffer serialisation/deserialisation, uplink and downlink ... 72
4.2.7. Backend NodeJS Server ... 75
4.2.8. ... 80
5. Πειράματα και αξιολόγηση ... **80**

6. Συμπεράσματα και μελλοντική εργασία 80
Αναφορές..... 81

1. Εισαγωγή

1.1. Κίνητρα

Industry 4.0

"Τέταρτη Βιομηχανική Επανάσταση", "4IR" ή "Industry 4.0" είναι λέξεις-κλειδιά και νεολογισμοί που περιγράφουν την γρήγορη τεχνολογική πρόοδο στον 21ο αιώνα. Ο όρος δημοσιοποιήθηκε το 2016 από τον Klaus Schwab, τον ιδρυτή και εκτελεστικό πρόεδρο του Παγκόσμιου Οικονομικού Φόρουμ, ο οποίος υποστηρίζει ότι οι αλλαγές δείχνουν μια σημαντική μετατόπιση στον βιομηχανικό καπιταλισμό.

Μέρος αυτής της φάσης της βιομηχανικής αλλαγής είναι η ένωση τεχνολογιών όπως η τεχνητή νοημοσύνη, η επεξεργασία γονιδίων και τα προηγμένα ρομπότ, που θολώνουν τα όρια μεταξύ των φυσικών, ψηφιακών και βιολογικών κόσμων.

Μέσα σε αυτό, λαμβάνουν χώρα θεμελιώδεις αλλαγές στον τρόπο λειτουργίας του παγκόσμιου δικτύου παραγωγής και εφοδιασμού μέσω της συνεχιζόμενης αυτοματοποίησης των παραδοσιακών κατασκευαστικών και βιομηχανικών πρακτικών, χρησιμοποιώντας σύγχρονη έξυπνη τεχνολογία, επικοινωνία μεγάλης κλίμακας μηχανής προς μηχανή (M2M) και το Διαδίκτυο των Πραγμάτων (IoT). Αυτή η ενσωμάτωση οδηγεί σε αυξανόμενη αυτοματοποίηση, βελτίωση της επικοινωνίας και αυτοπαρακολούθηση, καθώς και τη χρήση έξυπνων μηχανών που μπορούν να αναλύσουν και να διαγνώσουν προβλήματα χωρίς την ανάγκη ανθρώπινης παρέμβασης.

Αντιπροσωπεύει επίσης μια κοινωνική, πολιτική και οικονομική μετατόπιση από την ψηφιακή εποχή των τελευταίων δεκαετιών του 1990 και των αρχών του 2000 σε μια εποχή ενσωματωμένης συνδεσιμότητας που χαρακτηρίζεται από την πανταχού παρουσία της τεχνολογίας στην κοινωνία (δηλαδή ένα μετασύμπαν) που αλλάζει τους τρόπους με τους οποίους οι άνθρωποι βιώνουν και γνωρίζουν τον κόσμο γύρω τους. Υποστηρίζει ότι έχουμε δημιουργήσει και εισερχόμαστε σε μια ενισχυμένη κοινωνική πραγματικότητα σε σύγκριση με τις φυσικές αισθήσεις και τη βιομηχανική ικανότητα των ανθρώπων μόνο.

Στη τέταρτη βιομηχανική επανάσταση και στην αγρονομία ακριβείας γίνεται εκτεταμένη χρήση ασύρματων δικτύων σε συνδυασμό με έξυπνων αισθητήρων και όργανα μετρήσεων. Έξυπνοι αισθητήρες είναι συσκευές που παράγουν δεδομένα και επιτρέπουν περαιτέρω λειτουργίες από αυτοπαρακολούθηση και αυτορρύθμιση μέχρι παρακολούθηση της κατάστασης σύνθετων διαδικασιών. Με τη δυνατότητα ασύρματης επικοινωνίας, μειώνουν σημαντικά την προσπάθεια εγκατάστασης και βοηθούν στην πραγματοποίηση πυκνής διάταξης αισθητήρων.

Η αγρονομία ακριβείας είναι μια στρατηγική διαχείρισης της γεωργίας που βασίζεται στην παρατήρηση, τη μέτρηση και την αντίδραση στην χρονική και χωρική μεταβλητότητα προκειμένου να βελτιωθεί η βιωσιμότητα της γεωργικής παραγωγής. Χρησιμοποιείται τόσο στην παραγωγή καλλιεργειών όσο και στην εκτροφή ζώων. Η αγρονομία ακριβείας συχνά χρησιμοποιεί τεχνολογίες για την αυτοματοποίηση γεωργικών λειτουργιών, βελτιώνοντας τη διάγνωση, τη λήψη αποφάσεων ή την εκτέλεση των εργασιών.

Ο στόχος της έρευνας στην αγρονομία ακριβείας είναι να ορίσει ένα σύστημα υποστήριξης αποφάσεων για τη διαχείριση ολόκληρης της αγροκτηματικής μονάδας, με στόχο τη βέλτιστη εκμετάλλευση των εσόδων από τις εισροές ενώ προστατεύονται οι πόροι.

Η ανάπτυξη της αγρονομίας ακριβείας έχει διευκολυνθεί από την εισαγωγή του GPS και του GNSS, επιτρέποντας στον αγρότη ή τον ερευνητή να προσδιορίζει την ακριβή θέση του στο χωράφι. Αυτό δίνει τη δυνατότητα δημιουργίας χαρτών με τη χωρική μεταβλητότητα πολλών μεταβλητών, όπως παραγωγή καλλιεργειών, χαρακτηριστικά τοπογραφίας, περιεκτικότητα οργανικής ύλης, επίπεδα υγρασίας, επίπεδα αζώτου, pH, EC, Mg, K και άλλα. Η αγρονομία ακριβείας έχει επίσης διευκολυνθεί από αναπτυξιακές τεχνολογίες που επιτρέπουν τη χρήση αισθητήρων πραγματικού χρόνου απευθείας στο έδαφος, τα οποία μπορούν να μεταδίδουν δεδομένα ασύρματα χωρίς την παρουσία ανθρώπου. Μια απο τις επιλογές για ασύρματη τεχνολογία επικοινωνίας των αισθητήρων είναι το LoRaWAN.

Επικοινωνία με LoRa WAN και Protocol Buffers

Ένας από τους κύριους τομείς στον οποίο πιστεύω ότι μπορεί να υπάρξει αλλαγή είναι η σειριοποίηση των δεδομένων που μεταφέρονται δια μέσω των δικτύων LoRa WAN. Η πρόταση μου σε αυτόν τον τομέα είναι η χρήση των protocol buffers [1]. Τα protocol buffers είναι σχεδιασμένα έτσι ώστε να είναι αποδοτικά σε άποψη χώρου αλλά και ταχύτητας. Η μικρή δυαδική μορφή τους ελαχιστοποιεί το μέγεθος των σειριοποιημένων δεδομενων, το οποίο είναι πολύ βολικό για την χρήση τους με κανάλια επικοινωνίας μικρού bandwidth όπως και το LoRa WAN. Αυτή η αποδοτικότητα οδηγεί στην ταχύτερη μεταφορά των μηνυμάτων και καλύτερη ενεργειακή απόδοση σε περιπτώσεις που οι συσκευή που κάνει την σειριοποίηση χρησιμοποιεί μπαταρίες ως την πηγή ενέργειας της.

Προσφέρουν cross-language support το οποίο σημαίνει ότι η ίδια δομή δεδομένων έχουμε σχεδιάσει μπορει να χρησιμοποιηθεί με οποιαδήποτε γλώσσα προγραμματισμού επιλέξουμε. Είναι ένα αρκετά σημαντικό προτέρημα διότι σε ετερογενή περιβάλλοντα διαφορετικά κομμάτια του συστήματος μπορεί να δουλεύουν με διαφορετικές γλώσσες προγραμματισμού.

Ένα ακόμα προτέρημα των protocol buffers είναι ότι προσφέρουν επεκτασιμότητα (scalability), και συμβατότητα προς τα πίσω αλλά και προς τα μπροστά, τα οποία είναι όλα πολύ σημαντικά χαρακτηριστικά για μακροχρόνια προτζεκτ και εφαρμογές που αναπτύσσονται συνεχόμενα. Επιτρέπεται δηλαδή η προσθήκη νέων μεταβλητών πεδίων στην ήδη υπάρχουσα δομή χωρίς να σταματάει η λειτουργία της συσκευής που δεν έχει αναβαθμιστεί ακόμα αλλα ούτε και της συσκευής με την πιο πρόσφατη έκδοση της δομής δεδομένων.

Ο αυστηρός τρόπος ορισμού των του σχήματος(strictly defined schema) εξασφαλίζει την ακεραιότητα των δεδομένων κατα την σειριοποίηση και την αποσειριοποίηση και η ισχυρή ταυτοποίηση(strongly typed) των μεταβλητών πεδίων βοηθά στο να βρεθούν και να διορθωθούν σφάλματα κατά την περίοδο του σχεδιασμού, έτσι μειώνοντας πιθανά προβλήματα θα μπορούσαν να εμφανιστούν αργότερα, αυξάνοντας έτσι την αξιοπιστία του συστήματος.

Η υιοθέτηση και η χρήση των protocol buffers είναι αρκετα εύκολη λόγω της ξεκάθαρης σύνταξης για τον ορισμό του σχήματος. Κατά το compilation του σχήματος

δημιουργείται αυτόματα κώδικας για την σειριοποίηση και αποσειριοποίηση μειώνοντας τον κώδικα που θα πρέπει να γράψει ο προγραμματιστής επιτρέποντας του να συγκεντρωθεί στην ανάπτυξη της βασικής λειτουργίας. Επίσης ώντας λογισμικό της google έχει εκτεταμένο και καλογραμμένο documentation όπως επίσης και βοήθεια από την κοινότητα.

1.2. Δομή της εργασίας

Στο πρώτο κεφάλαιο θα μιλήσουμε εισαγωγικά για την πτυχιακή μιλώντας για κάποιες βασικές πληροφορίες και τεχνολογίες που θα χρειαστούν για την υλοποίηση όπως επίσης και για τα κίνητρα μου για την πτυχιακή.

Στο δεύτερο κεφάλαιο θα μιλήσουμε και θα αναλύσουμε τις state of the art τεχνολογίες που χρησιμοποιήθηκαν στην δημιουργία της εργασίας. Θα αναλύσουμε διάφορες τεχνολογίες όπως απομακρυσμένη επικοινωνία, σειριοποίηση δεδομένων και άλλα.

Στο τρίτο κεφάλαιο θα δούμε πως έγινε η σχεδίαση του όλου συστήματος και ποιές ήταν η απαιτήσεις οι οποίες έπρεπε να έχουμε υπ όψιν.

Στο τέταρτο κεφάλαιο θα δούμε πώς έγινε η υλοποίηση του hardware και του software του συστήματος, δείχνοντας τα εξαρτήματα και τις βιβλιοθήκες που χρησιμοποιήθηκαν.

Στο τελευταίο κεφάλαιο μιλάμε για το αποτέλεσμα της εργασίας καθώς και δίνοντας ιδέες για μελλοντική εργασία.

2. State of the art

2.1. Internet Of Things γενικά

Το Internet of Things [8] περιγράφει συσκευές με αισθητήρες, δυνατότητα επεξεργασίας, λογισμικό και άλλες τεχνολογίες που συνδέονται και ανταλλάσουν δεδομένα με άλλες συσκευές και συστήματα μέσω του ίντερνετ ή άλλα δίκτυα επικοινωνίας. Το Internet of Things συμπεριλαμβάνει ηλεκτρονικά, επικοινωνίες και μηχανική επιστήμης υπολογιστών. Το όνομα Internet of Things έχει θεωρηθεί ως λάθος χαρακτηρισμός διότι οι συσκευές δεν χρειάζεται να είναι συνδεδεμένες στο δημόσιο διαδίκτυο, χρειάζονται απλά να είναι συνδεδεμένες σε ένα δίκτυο και να μπορεί κάποιος ή κάτι να απευθυνθεί σε αυτές.

Ο τομέας έχει αναπτυχθεί λόγω του ότι πολλές τεχνολογίες όπως το ubiquitous computing(πανταχού παρούσα υπολογιστική) [9], εμπορικοί αισθητήρες, και όλο πιο δυνατά ενσωματωμένα συστήματα όπως επίσης και μηχανική μάθηση. Παλιότεροι τομείς όπως τα ενσωματωμένα, ασύρματα δίκτυα αισθητήρων, συστήματα ελέγχου και ο αυτοματισμός καθιστούν δυνατό το Internet of Things ξεχωριστά αλλά και συλλογικά. Στην καταναλωτική αγορά η τεχνολογία του IoT είναι συνώνυμη με τον ορισμό smart home(έξυπνο σπίτι) το οποίο αναφέρεται σε οικιακές συσκευές όπως φωτιστικά, θερμοστάτες και άλλες συσκευές που υποστηρίζουν ένα ή και παραπάνω οικοσυστήματα ελέγχου που μπορούν να ελεγχθούν από άλλες συσκευές όπως κινητά και συσκευές εικονικών βοηθών όπως το Amazon Echo [10] και το Google Nest [11].

Ενώ ο όρος «Διαδίκτυο των Πραγμάτων» είναι σχετικά νέος, η έννοια του συνδυασμού υπολογιστών και δικτύων για την παρακολούθηση και τον έλεγχο συσκευών υπάρχει εδώ και δεκαετίες. Μέχρι τα τέλη της δεκαετίας του 1970, για παράδειγμα, συστήματα για την απομακρυσμένη παρακολούθηση μετρητών στο ηλεκτρικό δίκτυο μέσω τηλεφωνικών γραμμών ήταν ήδη σε εμπορική χρήση [12]. Στη δεκαετία του 1990, οι εξελίξεις στην ασύρματη τεχνολογία επέτρεψαν τις λύσεις «μηχανή-προς-μηχανή» (Machine 2 Machine, M2M) για την επιχείρηση και τη βιομηχανία για την παρακολούθηση και τη λειτουργία εξοπλισμού να γίνουν ευρέως διαδεδομένες. Πολλές από αυτές τις πρώιμες λύσεις M2M, ωστόσο, βασίζονταν σε κλειστά δίκτυα ειδικού σκοπού και σε ιδιόκτητα ή βιομηχανικά πρότυπα, παρά σε δίκτυα που βασίζονται στο Πρωτόκολλο Διαδικτύου (IP) και στα πρότυπα του Διαδικτύου.



Fig. 3 Internet of Things(source: https://its.ucsc.edu/news/internet-of-things.html)

Υπάρχουν πολλές ανησυχίες όσον αφορά τα ρίσκα στην ανάπτυξη των τεχνολογιών IoT, ειδικά στους τομείς της ιδιωτικότητας και την ασφάλεια το οποίο έχει οδηγήσει τις κυβερνήσεις να πάρουν μέτρα, όπως την δημιουργία διεθνών και τοπικών στανταρ, οδηγούς και ρυθμιστικά πλαίσια. Λόγω της διασυνδεδεμένης φύσης τους, οι IoT συσκευές είναι ευάλωτες σε παραβιάσεις ασφαλείας και ανησυχίες ως πρός την ιδιωτικότητα.

2.2. Αισθητήρες

Τι είναι οι αισθητήρες;

Οι αισθητήρες είναι ηλεκτρονικές συσκευές που μπορούν να μετρήσουν επιθυμητές πληροφορίες από το περιβάλλον, όπως θερμοκρασία, υγρασία, πίεση, επίπεδο φωτεινότητας, και να τις μετατρέψουν σε ηλεκτρικά σήματα τα οποία μπορούν να διαβάσουν οι άνθρωποι ή μηχανές για αποθήκευση και επεξεργασία. Δουλεύουν ανιχνεύοντας φυσικές ή χημικές αλλαγές στο περιβάλλον τους.

Μερικοί αισθητήρες στην γεωργία.

Διηλεκτρικοί αισθητήρες υγρασίας [5]. Είναι αισθητήρες υγρασίας οι οποίοι χρησιμοποιούν τον διηλεκτρικό συντελεστή για να μετρήσουν το επίπεδο υγρασίας στο χώμα. Για παράδειγμα ο συντελεστής για ξηρό χώμα είναι 3 με 5, για αέρα 1 και για το νερό 80 [4]. Αυτοί οι αισθητήρες χρησιμοποιούνται για την βέλτιστη ανάπτυξη καλλιέργειας.

Ηλεκτροχημικοί αισθητήρες [5] τοποθετούνται σε ειδικά έλκηθρα και συλλέγουν δεδομένα τα οποία βοηθούν στην χαρτογράφηση χημικών στο χώμα. Δίνουν ακριβείς μετρήσεις για τα θρεπτικά στοιχεία και τα επίπεδα pH του εδάφους τα οποια είναι σημαντικά για την γεωργία. Τα αποτελέσματα των μετρήσεων δίνονται σε ειδικά εργαστήρια που εξειδικεύονται στην ανάλυση εδάφους και πραγματοποιούνται τεστ. Για μετρήσεις επιπέδου pH χωρίς σφάλματα χρησιμοποιούνται ιοντοεπιλεκτικά ηλεκτρόδια. Τα ηλεκτρόδια αυτα μετρούν τις κινήσεις ιόντων οπως το υδρογόνο το νιτρικό άλας και το κάλιο.

Οι αισθητήρες αυτοί μπορούν επίσης να παρακολουθήσουν την ανάπτυξη των φυτών, τις ασθένειες και την περιβαλλοντική μόλυνση [6]. Ορισμένοι αισθητήρες βοηθάνε στο να εντοπίζουν πτητικές οργανικές ενώσεις στο έδαφος. Τέτοιοι αισθητήρες είναι συνήθως φτιαγμένοι από νανοσωλήνες άνθρακα, γραφίτη, μειωμένο οξείδιο γραφίτη και νανοσωματίδια χρυσού. Άλλοι τέτοιοι αισθητήρες έχουν φτιαχτεί για να εντοπίζουν φυτικό χυμό(plant sap) και άζωτο στο έδαφος.

Αισθητήρες θέσεις, ή αλλιώς αγροτικοί σταθμοί καιρού με GPS, τοποθετούνται σε διάφορα μέρη μέσα στα χωράφια. Αυτοί οι αισθητήρες ακριβείας χρησιμοποιούνται για να γίνουν μετρήσεις απόστασης και ύψους κάποιου σημείου σε μια απαιτούμενη περιοχή.

Περιορισμοι αισθητήρων.

Κόστος: Οι πιο προηγμένοι αισθητήρας που χρησιμοποιούνται έχουν μεγάλο κόστος, αυτό μπορεί να είναι απαγορευτικό για μικρούς αγρότες και οικογενειακές επιχειρήσεις. Ένας λόγος που ενθαρρύνει τους αγρότες από το να υιοθετούν τέτοιου είδους τεχνολογίες είναι το αρχικό κόστος αγοράς και εγκατάστασης τέτοιων συστημάτων, ασχέτως των πλεονεκτημάτων που προσφέρουν στην αποδοτικότητα και την παραγωγικότητα.

Συντήρηση και ανθεκτικότητα: Οι αισθητήρες στην γεωργία λειτουργούν και εκτίθενται σε πολύ σκληρό περιβάλλον και συνθήκες, όπως γη υγρασία, η σκόνη, και οι ακραίες θερμοκρασίες. Όλα αυτά μπορούν να επηρεάσουν την ακρίβεια των αισθητήρων και να μειώσουν την διάρκεια ζωής τους.

Ακρίβεια: Η ακρίβεια των αισθητήρων μπορεί να έχει μεγάλες διαφορές από κατασκευαστή σε κατασκευαστή, από την τοποθεσία εγκατάστασης και τις περιβαλλοντικές συνθήκες. Ακόμα και μικρές αποκλίσεις σε μετρήσεις μπορούν να έχουν σημαντικές επιπτώσεις στις αποφάσεις που θα παρθούν για την καλλιέργεια. Οι αισθητήρες επίσης πρέπει να παρέχουν αξιοπιστία και συνεπή δεδομένα για να είναι χρήσιμοι για τους αγρότες.

Συμβατότητα: Διαφορετικοί κατασκευαστές μπορεί να χρησιμοποιούν διαφορετικά πρωτόκολλα επικοινωνίας το οποίο μπορεί να δημιουργήσει προβλήματα συμβατότητα. Αυτό μπορεί να δυσκολέψει την ενσωμάτωση των συστημάτων και να αυξήσει την πολυπλοκότητα της διαχείριση των δεδομένων.

2.3. Ασύρματα δίκτυα αισθητήρων(WSN)

Τα ασύρματα δίκτυα αισθητήρων είναι δίκτυα που αποτελούνται από αισθητήρες οι οποίοι είναι διασκορπισμένοι σε έναν χώρο, παρακολουθούν το περιβάλλον τους και στέλνουν τα δεδομένα που έχουν μαζέψει σε μία κεντρική τοποθεσία. Μερικές δυνατότητες των WSN είναι να μετρούν περιβαλλοντικές συνθήκες όπως η θερμοκρασία, το ήχος, τα επίπεδα ρύπανσης, η υγρασία και ο άνεμος. Τα δίκτυα αυτά μοιάζουν με τα δίκτυα ad hoc στο ότι

βασίζονται στην ασύρματη συνδεσιμότητα και τον τυχαίο σχηματισμό δικτύων, ώστε τα δεδομένα των αισθητήρων να μπορούν να μεταφέρονται ασύρματα.

Τα σύγχρονα δίκτυα είναι αμφίδρομα, συλλέγοντας δεδομένα και δίνοντας τον έλεγχο της δραστηριότητας και την παραμετροποίηση των αισθητήρων. Στρατιωτικές εφαρμογές όπως η παρακολούθηση του πεδίου μάχης είναι ένας από τους κύριους λόγους αναπτυξης αυτων των δικτύων [7]. Τέτοια δίκτυα βρίσκουν χρήση σε βιομηχανικές εγκαταστάσεις και καταναλωτικές εφαρμογές, όπως η παρακολούθηση και ο έλεγχος βιομηχανικών διαδικασιών και η παρακολούθηση της υγείας των μηχανών αλλά και στη γεωργία. Ένα WSN Είναι φτιαγμένο από μερικούς εκατοντάδες έως και χιλιάδες κόμβους, Όπου κάθε κόμβος είναι συνδεδεμένος σε άλλους αισθητήρες.

Κάθε τέτοιος κόμβος συνήθως έχει πολλά κομμάτια, ένα ποδέκτη radio με ενσωματωμένη κεραία ή σύνδεση σε μία εξωτερική κεραία, έναν μικροελεγκτή, ένα ηλεκτρονικό κύκλωμα το οποίο επικοινωνεί με τους αισθητήρες και μία πηγή ενέργειας που συνήθως είναι μια μπαταρία ή μία ενσωματωμένη μορφή συλλογής ενέργειας. Ένας κόμβος αισθητήρα μπορεί να ποικίλει σε μέγεθος από ένα κουτί παπουτσιών μέχρι και μια συσκευή μικρότερη από ένα κινητό. Το κόστος του κόμβου είναι επίσης μεταβλητό κυμαινόμενο από λίγα έως εκατοντάδες ευρώ ανάλογα με την πολυπλοκότητα του κόμβου. Οι περιορισμοί σε μέγεθος και κόστος περιορίζουν τους πόρους όπως η ενέργεια, η μνήμη, η ταχύτητα επεξεργασίας, και το εύρος ζώνης επικοινωνιών. Η τοπολογία ενός WSN Μπορεί να πάρει τη μορφή ενός απλού δικτύου αστέρα Αλλά μέχρι και ενός προηγμένου multi-hop ασύρματο δίκτυο πλέγματος.

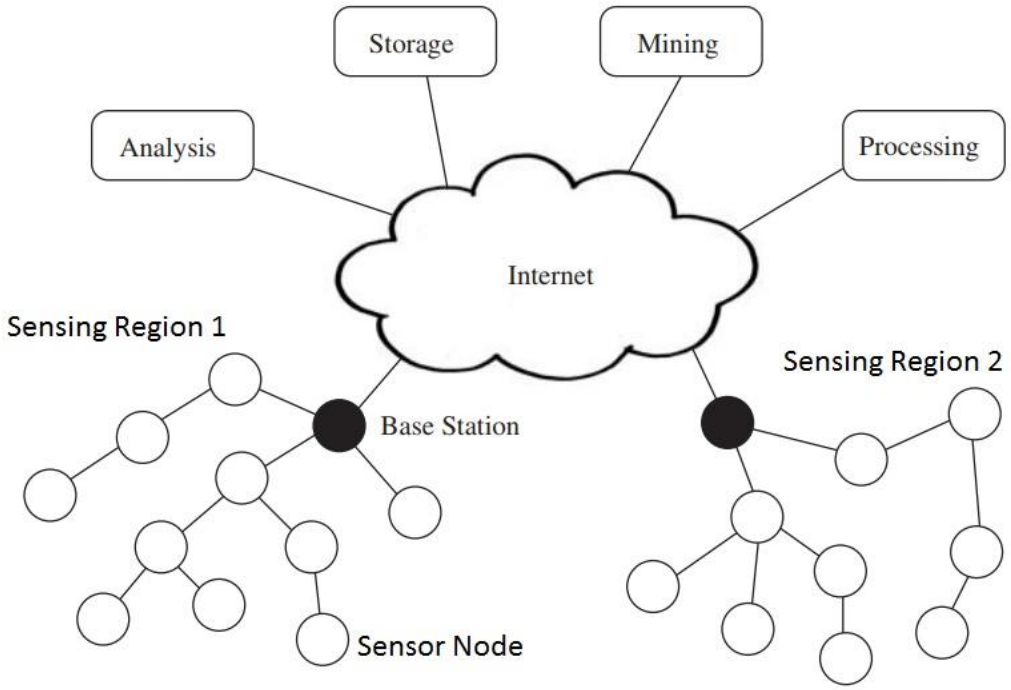


Fig. 1 WSN δίκτυο (source: https://www.electronicshub.org/wireless-sensor-networks-wsn/)

2.4. Internet Of Things στην γεωργία

Το IoT στην γεωργία, και ειδικότερα στον τομέα της παρακολούθησης της υγείας των φυτών, προσφέρει σημαντικές δυνατότητες για την συλλογή, αποθήκευση και ανάλυση πληροφοριών από τους αισθητήρες. Αυτές οι τεχνολογίες μπορούν να βοηθήσουν τους αγρότες να βελτιώσουν την αποδοτικότητα και την παραγωγικότητα των καλλιεργειών τους, ενώ παράλληλα συμβάλλουν στην αειφόρο ανάπτυξη. Ακολουθεί μια περιγραφή του πώς το IoT μπορεί να χρησιμοποιηθεί σε αυτό το πλαίσιο:

Μετρήσεις για τη γεωργία ακριβείας

Η χρήση των WSN στην γεωργία ακριβείας εφαρμόζεται σε πολλά πράγματα, όπως την παρακολούθηση του περιβάλλοντος, υπαίθρια παρακολούθηση, παρακολούθηση θερμοκηπίων, διαχείριση συστημάτων ύδρευσης και άρδευσης και άλλα. Η πιο συνήθης χρήση τους στην γεωργία είναι η παρακολούθηση του περιβάλλοντος και διάφορες παραμέτρους του, όπως η υγρασία, η θερμοκρασία, το pH και η ηλεκτρική αγωγιμότητα του χώματος. Το σύστημα παρακολούθησης μπορεί να βασίζεται στην ασύρματη τεχνολογία zigbee ή όπως στην περίπτωση μας στο WiFi και το LoRa WAN. Όλοι αυτοί οι παράμετροι είναι σημαντικοί για την γεωργία ακριβείας.

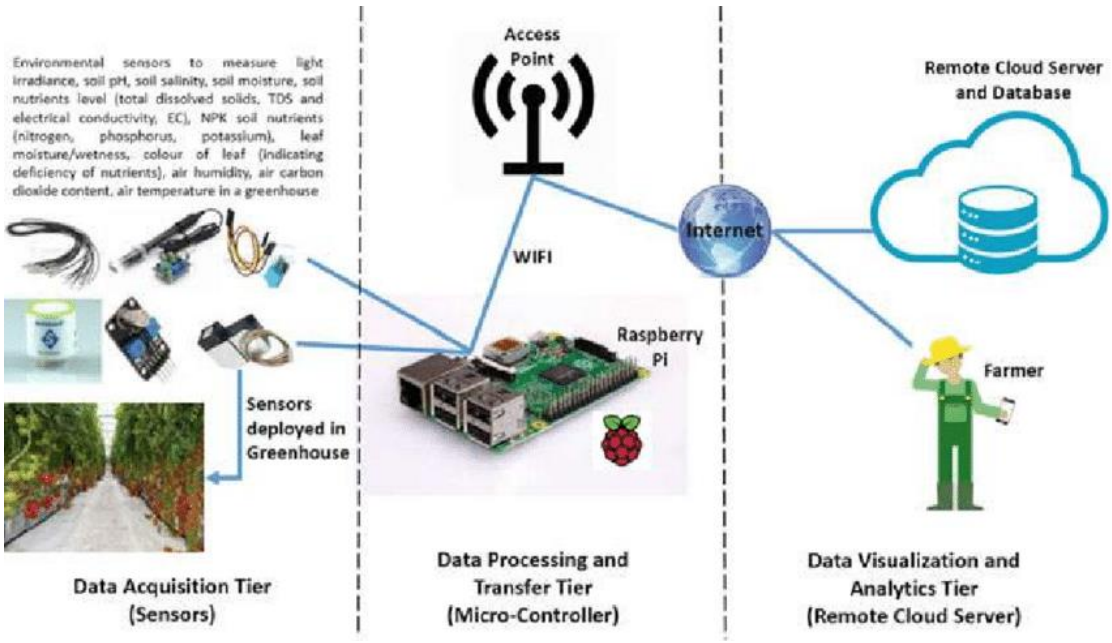


Fig. 2 WSN δίκτυο στην γεωργία ακρίβειας(source: https://www.researchgate.net/figure/High-level-WSN-architecture-for-precision-agriculture_fig1_330149051)

Διαχείριση άρδευσης

Η άρδευση είναι μια απαραίτητη πρακτική σε πολλά γεωργικά συστήματα καλλιέργειας σε άνυδρες περιοχές, όπου η γεωργική παραγωγή είναι χαμηλή και το νερό είναι ένας σπάνιος και ζωτικός πόρος. Η κατάλληλη άρδευση είναι κρίσιμη για τη βελτίωση της απόδοσης των καλλιεργειών. Η παραδοσιακή άρδευση των χωραφιών συχνά απαιτεί ανθρώπινη εργασία και χρειάζεται τεράστια ανθρώπινα και υλικά μέσα. Τα μπορικά συστήματα ελέγχου άρδευσης είναι συνήθως ακριβά και δεν είναι αποτελεσματικά στη διαχείριση των σπάνιων υδάτινων πόρων. Από την άλλη, η διαχείριση άρδευσης με τη χρήση ασύρματων δικτύων αισθητήρων (WSN) μπορεί να εξασφαλίσει καλύτερη απόδοση

καλλιεργειών καλής ποιότητας και να αποδεχτεί οποιαδήποτε επιθυμητή στρατηγική προγραμματισμού άρδευσης για να ανταποκριθεί στις συγκεκριμένες περιβαλλοντικές απαιτήσεις. Τα WSN μπορούν να εφαρμοστούν ως εργαλείο για την κατάλληλη τακτική άρδευσης ώστε να βελτιωθεί η απόδοση των καλλιεργειών.

Το IoT στην Παρακολούθηση της Υγείας των Φυτών

Η χρήση του IoT στη γεωργία επιτρέπει την συνεχόμενη παρακολούθηση της υγείας των φυτών μέσω δικτυωμένων αισθητήρων που εγκαθίστανται στους αγρούς. Αυτοί οι αισθητήρες μπορούν να μετρούν διάφορες παραμέτρους, όπως την υγρασία του εδάφους, τη θερμοκρασία, την υγρασία του αέρα, την έκθεση στο φως, και τη συγκέντρωση θρεπτικών ουσιών.

Συλλογή Δεδομένων

Οι αισθητήρες συλλέγουν δεδομένα σε πραγματικό χρόνο και τα αποστέλλουν σε μια κεντρική πλατφόρμα μέσω ασύρματων δικτύων, όπως το LoRaWAN. Αυτό επιτρέπει την συνεχή παρακολούθηση των συνθηκών στο πεδίο χωρίς την ανάγκη για φυσική παρουσία του αγρότη. Τα δεδομένα που συλλέγονται περιλαμβάνουν κρίσιμες πληροφορίες για την υγεία των φυτών και την ποιότητα του εδάφους.

Αποθήκευση Δεδομένων

Τα δεδομένα που συλλέγονται από τους αισθητήρες αποθηκεύονται σε μια κεντρική βάση δεδομένων. Αυτή η βάση δεδομένων μπορεί να είναι τοπική ή να φιλοξενείται στο cloud, εξασφαλίζοντας την ασφαλή και αξιόπιστη αποθήκευση μεγάλου όγκου δεδομένων. Η χρήση πρωτοκόλλων όπως το Protocol Buffers επιτρέπει την αποδοτική και συμπιεσμένη αποθήκευση των δεδομένων, μειώνοντας τις απαιτήσεις χώρου και βελτιώνοντας την ταχύτητα μετάδοσης.

Ανάλυση Δεδομένων

Η ανάλυση των δεδομένων πραγματοποιείται μέσω εξειδικευμένων λογισμικών που μπορούν να επεξεργαστούν τα δεδομένα και να παρέχουν πολύτιμες πληροφορίες για την υγεία των φυτών. Αυτή η ανάλυση μπορεί να περιλαμβάνει τη δημιουργία γραφημάτων και αναφορών που απεικονίζουν τις αλλαγές στις περιβαλλοντικές συνθήκες, την ανίχνευση ανωμαλιών ή προβλημάτων υγείας στα φυτά, και την πρόβλεψη των μελλοντικών αναγκών των καλλιεργειών.

Χρήση των Πληροφοριών

Οι πληροφορίες που προκύπτουν από την ανάλυση των δεδομένων μπορούν να χρησιμοποιηθούν για την λήψη ενημερωμένων αποφάσεων σχετικά με τη διαχείριση των

καλλιεργειών. Οι αγρότες μπορούν να προσαρμόσουν τις πρακτικές άρδευσης, λίπανσης και φυτοπροστασίας με βάση τις πραγματικές ανάγκες των φυτών, βελτιστοποιώντας την χρήση των πόρων και μειώνοντας τις περιβαλλοντικές επιπτώσεις. Επιπλέον, η έγκαιρη ανίχνευση προβλημάτων υγείας μπορεί να βοηθήσει στην άμεση αντιμετώπισή τους, προλαμβάνοντας την εξάπλωση ασθενειών και την απώλεια παραγωγής.

2.5. Ήδη υπάρχοντα συστήματα της αγοράς

Μια εταιρεία που ειδικεύεται στα συστήματα παρακολούθησης και ελέγχου θερμοκηπίων είναι η Priva [23]. Η Priva διαθέτει διάφορα συστήματα όπως σύστημα ελέγχου κλίματος για θερμοκήπια, σύστημα άρδευσης, και σύστημα ασύρματων αισθητήρων. Ολα τα συστήματα αυτά ενσωματώνονται μεταξύ τους για την συλλογή δεδομένων των φυτών για τον έλεγχο του κλίματος του θερμοκηπίου για την καλύτερη δυνατή απόδοση και παραγωγή.

Το σύστημα ασύρματων αισθητήρων της Priva κάνει παρακολούθηση της θερμοκρασίας, της υγρασίας, του επιπέδου του διοξειδίου άνθρακα και ένταση φωτός. Οι αισθητήρες χρησιμοποιούν την ασύρματη τεχνολογία της Aranet [24], η Aranet Radio είναι ασύρματη τεχνολογία σχεδιασμένη για αποτελεσματική και αξιόπιστη παρακολούθηση σε διάφορα περιβάλλοντα, όπως τα θερμοκήπια, τη γεωργία και βιομηχανικούς χώρους. Το Aranet Radio χρησιμοποιεί διαμόρφωση LoRa αλλά όχι το LoRaWAN protocol. Το Aranet radio λειτουργεί στις ISM 868/929 MHz συχνότητες. Είναι απομονωμένο πρωτόκολλο επικοινωνίας που σημαίνει ότι δεν είναι κομμάτι του LoRaWAN και δεν υπάρχουν παρεμβολές μεταξύ τους.

Η Aranet υποστηρίζει ότι η σταθερή τιμή του spread factor εξασφαλίζει προβλέψιμα αποτελέσματα. Η μονόδρομη επικοινωνία είναι βέλτιστη για την περίπτωση χρήσης και βοηθάει στην μείωση της κατανάλωσης ενέργειας και αποκλείει πιθανές επιθέσεις στους αισθητήρες. Το μονόδρομο πρωτόκολλο αφαιρεί την δυνατότητα μετάδοσης δεδομένων πριν από μια προκαθορισμένη ώρα, κάτι που δεν είναι ουσιώδες, καθώς το διάστημα χρόνου κάθε μέτρησης για τους αισθητήρες Aranet μπορούν να οριστούν σε μόλις 1 λεπτό, σε σύγκριση με τα τυπικά διαστήματα χρόνου μέτρησης LoRaWAN που μπορεί να είναι 10, 30 ή και 60 λεπτών.

Το μικρότερο μέγεθος πακέτου δεδομένων εξοικονομεί χρόνο εκπομπής ή αλλιώς time on air, εξοικονομεί τους πόρους της μπαταρίας και ελαχιστοποιεί την πιθανότητα απώλειας δεδομένων λόγω σύγκρουσης με άλλες εκπομπές αισθητήρων στο σύστημα ή παρεμβολές από εξωτερικούς πομπούς. Το μικρότερο μέγεθος πακέτου δεδομένων σημαίνει επίσης ότι αυτά τα σήματα δεν είναι ελκυστικά για να αποτελέσουν μέρος τυχαίων επιθέσεων.

Το gateway στην περίπτωση της Aranet δεν είναι ένα απλό gateway όπως στο LoRaWAN αλλά λειτουργεί και ως έναν webserver ο οποίος συλλέγει τα δεδομένα των αισθητήρων με δυνατότητα να αποθηκεύσει δεδομένα 10 ετών. Σε περίπτωση που η σύνδεση στο διαδίκτυο είναι αδύνατη τότε τα δεδομένα των αισθητήρων θα μείνουν στο gateway μέχρι να επανέλθει η σύνδεση στο διαδίκτυο και να σταλούν στο cloud σύστημα τους.

2.6. RS-485

Το RS-485, γνωστό επίσης ως TIA-485(-A) ή EIA-485, είναι ένα στάνταρ, που εισήχθη αρχικά το 1983, το οποίο ορίζει τα ηλεκτρικά χαρακτηριστικά των οδηγών(drivers) και δεκτών για χρήση σε συστήματα σειριακής επικοινωνίας. Τα ηλεκτρικά σήματα είναι ισορροπημένα, και υποστηρίζει συστήματα πολλών σημείων. Το στανταρ δημοσιεύτηκε από κοινού από την Ένωση Βιομηχανίας Τηλεπικοινωνιών και την Ένωση Ηλεκτρονικών Βιομηχανιών (TIA/EIA). Ένα μεγάλο πλεονέκτημα των δικτύων που υλοποιούν το στανταρ είναι ότι αποτελεσματικά σε μεγάλες αποστάσεις αλλά και σε ηλεκτρικά θορυβώδη περιβάλλοντα. Πολλαπλοί δέκτες μπορούν να συνδεθούν σε ένα τέτοιο δίκτυο σε γραμμικό, πολυ-διακλαδισμένο δίαυλο(multidrop bus [14]). Αυτά τα χαρακτηριστικά καθιστούν το RS-485 χρήσιμο σε συστήματα βιομηχανικού ελέγχου και παρόμοιες εφαρμογές.

Το RS-485 υποστηρίζει φθηνά τοπικά δίκτυα και πολλαπλούς συνδέσμους επικοινωνίας(multidrop bus [14]), χρησιμοποιώντας την ίδια διαφορική σήμανση πάνω σε ζεύγη συνδεδεμένα καλώδια όπως το RS-422. Γενικά γίνεται δεκτό ότι το RS-485 μπορεί να χρησιμοποιηθεί με ρυθμούς μετάδοσης δεδομένων έως και 10 Mbit/s ή, σε χαμηλότερες ταχύτητες, αποστάσεις έως 1.200 μέτρα [15]. Ως κανόνας, η ταχύτητα σε bit/s πολλαπλασιασμένη με το μήκος σε μέτρα δεν θα πρέπει να υπερβαίνει τα 10^8. Έτσι, ένα καλώδιο 50 μέτρων δεν θα πρέπει να μεταδίδει σήματα με ταχύτητα μεγαλύτερη από 2 Mbit/s.

Σε αντίθεση με το RS-422, το οποίο έχει ένα κύκλωμα οδηγού(driver) που δεν μπορεί να απενεργοποιηθεί, οι οδηγοί του RS-485 χρησιμοποιούν λογική τριών καταστάσεων που επιτρέπει την απενεργοποίηση των μεμονωμένων πομπών. Αυτό επιτρέπει στο RS-485 να εφαρμόσει γραμμικές τοπολογίες δικτύου χρησιμοποιώντας μόνο δύο καλώδια. Το εξοπλισμό που βρίσκεται κατά μήκος ενός συνόλου καλωδίων RS-485 ονομάζεται ανταλλάξιμα κόμβοι, σταθμοί ή συσκευές.

Η συνιστώμενη διάταξη των καλωδίων είναι ως συνδεδεμένη σειρά από κόμβους σημείου-προς-σημείο (multi-dropped), δηλαδή μια γραμμή ή λεωφόρος, όχι αστέρι, κύκλος ή πολλαπλώς συνδεδεμένο δίκτυο. Οι τοπολογίες αστέρι και κύκλου δεν συνιστώνται λόγω ανακλάσεων σήματος ή υπερβολικά χαμηλής ή υψηλής αντίστασης τερματισμού. Αν μια διάταξη αστέρι είναι αναπόφευκτη, υπάρχουν διαθέσιμοι ειδικοί επαναλήπτες(repeaters) RS-485 οι οποίοι ακούν αμφίδρομα για δεδομένα σε κάθε έκταση και στη συνέχεια αναμεταδίδουν τα δεδομένα σε όλες τις άλλες εκτάσεις.

Ιδανικά, τα δύο άκρα του καλωδίου θα έπρεπε να έχουν έναν αντιστάτη τερματισμού συνδεδεμένο στα δύο καλώδια. Χωρίς αντιστάσεις τερματισμού, οι ανακλάσεις σήματος από το ατερμάτιστο άκρο του καλωδίου μπορούν να προκαλέσουν διαφθορά δεδομένων. Οι αντιστάσεις τερματισμού μειώνουν επίσης την ευαισθησία στον ηλεκτρικό θόρυβο λόγω της χαμηλότερης αντίστασης. Η τιμή κάθε αντίστασης τερματισμού πρέπει να είναι ίση με την χαρακτηριστική αντίσταση του καλωδίου (συνήθως, 120 ohm για τα πλεκτά ζεύγη).

Ο τερματισμός περιλαμβάνει επίσης αντιστάσεις pull-up και αντιστάσεις pull-down για την δημιουργία ασφαλούς μεροληψίας(fail-safe bias) για κάθε γραμμή δεδομένων στην περίπτωση που οι γραμμές δεν χρησιμοποιούνται από καμία συσκευή. Με αυτό τον τρόπο, οι γραμμές θα έχουν βάσεις σε γνωστές τάσεις και οι κόμβοι δεν θα ερμηνεύουν το θόρυβο από μη διαμορφωμένες γραμμές ως πραγματικά δεδομένα. Χωρίς αντιστάσεις πόλωσης, οι

γραμμές δεδομένων επηρεάζονται από ηλεκτρικό θόρυβο με τον μεγαλύτερο βαθμό όταν όλοι οι σταθμοί συσκευών είναι σιωπηλοί ή απενεργοποιημένοι.

Το στανταρ

Η ΕΙΑ παλιότερα είχε χαρακτηρίσει όλα τα πρότυπά της με το πρόθεμα "RS" (Συνιστώμενο Στανταρ), αλλά η ΕΙΑ-ΤΙΑ αντικατέστησε επίσημα το "RS" με το "ΕΙΑ/ΤΙΑ" για να βοηθήσει στον προσδιορισμό της προέλευσης των προτύπων της. Η ΕΙΑ έχει διαλυθεί επίσημα και το στανταρ διατηρείται τώρα από την ΤΙΑ ως ΤΙΑ-485, αλλά οι μηχανικοί και οδηγοί εφαρμογών συνεχίζουν να χρησιμοποιούν την ονομασία RS-485. Η αρχική έκδοση της ΕΙΑ RS-485 χρονολογείται τον Απρίλιο του 1983.

Το RS-485 καθορίζει μόνο τα ηλεκτρικά χαρακτηριστικά της γεννήτριας και του δέκτη: το φυσικό επίπεδο. Δεν καθορίζει ή προτείνει κανένα πρωτόκολλο επικοινωνίας. Άλλα πρότυπα καθορίζουν τα πρωτόκολλα για την επικοινωνία μέσω μιας σύνδεσης RS-485. Η εισαγωγή στο στανταρ αναφέρεται στο The Telecommunications Systems Bulletin TSB-89 το οποίο περιέχει οδηγίες εφαρμογής, συμπεριλαμβανομένου του ρυθμού σήμανσης δεδομένων έναντι του μήκους του καλωδίου, του μήκους του αποκλαδώματος και των διαμορφώσεων.

Το Τμήμα 4 καθορίζει τα ηλεκτρικά χαρακτηριστικά της γεννήτριας (πομπού ή οδηγού), του δέκτη, του πομποδέκτη και του συστήματος. Αυτά τα χαρακτηριστικά περιλαμβάνουν: τον ορισμό μιας μονάδας φορτίου, εύρος τάσης, τιμές τάσης ανοιχτού κυκλώματος, κατώτατες τιμές, και ανεκτικότητα στις προσωρινές αλλαγές. Καθορίζει επίσης τρία σημεία διεπαφής της γεννήτριας (γραμμές σήματος): A, B και C. Τα δεδομένα μεταδίδονται στο A και B. Το C είναι γείωση. Αυτή η ενότητα καθορίζει επίσης τη λογική καταστάτη 1 (απενεργοποίηση) και 0 (ενεργοποίηση), με την πολικότητα μεταξύ των τερματικών A και B. Εάν το A είναι αρνητικό σε σχέση με το B, η κατάσταση είναι δυαδικό 1. Η αντιστροφή πολικότητας (το A θετικό σε σχέση με το B) είναι δυαδικό 0. Το στανταρ δεν αναθέτει καμία λογική λειτουργία στις δύο καταστάσεις. Το RS-485 όπως και το RS-422, μπορεί να γίνει και full duplex χρησιμοποιώντας 4 καλώδια.

Εφαρμογές

Τα σήματα RS-485 χρησιμοποιούνται σε ένα ευρύ φάσμα συστημάτων υπολογιστών και αυτοματισμών. Σε ένα σύστημα υπολογιστή, το SCSI-2 και το SCSI-3 μπορούν να χρησιμοποιήσουν αυτή την προδιαγραφή για να υλοποιήσουν το φυσικό επίπεδο για τη μετάδοση δεδομένων μεταξύ ενός ελεγκτή και ενός σκληρού δίσκου. Το RS-485 χρησιμοποιείται για επικοινωνίες χαμηλής ταχύτητας δεδομένων στα οχήματα των καμπινών των εμπορικών αεροσκαφών. Απαιτεί ελάχιστη καλωδίωση και μπορεί να μοιράζεται την καλωδίωση μεταξύ πολλών θέσεων, μειώνοντας το βάρος.

Χρησιμοποιούνται σε προγραμματιζόμενους λογικούς ελεγκτές και στους βιομηχανικούς χώρους. Το RS-485 χρησιμοποιείται ως το φυσικό επίπεδο κάτω από πολλά τυποποιημένα και ιδιόκτητα πρωτόκολλα αυτοματισμού που χρησιμοποιούνται για την υλοποίηση βιομηχανικών συστημάτων ελέγχου, συμπεριλαμβανομένων των πιο κοινών εκδόσεων των Modbus και Profibus. Το DH 485 είναι ένα ιδιόκτητο πρωτόκολλο επικοινωνίας που χρησιμοποιείται από την Allen-Bradley στη σειρά των βιομηχανικών μονάδων ελέγχου τους. Χρησιμοποιώντας μια σειρά από ειδικές συσκευές διεπαφής, επιτρέπει στους

υπολογιστές και τους βιομηχανικούς ελεγκτές να επικοινωνούν. Επειδή είναι διαφορικό, αντιστέκεται στις ηλεκτρομαγνητικές παρεμβολές από κινητήρες και εξοπλισμό συγκόλλησης.

Σε θέατρα και χώρους παραστάσεων, τα δίκτυα RS-485 χρησιμοποιούνται για τον έλεγχο του φωτισμού και άλλων συστημάτων χρησιμοποιώντας το πρωτόκολλο DMX512. Το RS-485 χρησιμεύει ως φυσικό επίπεδο για τη διασύνδεση ψηφιακού ήχου AES3.

Το RS-485 χρησιμοποιείται επίσης στην αυτοματοποίηση κτιρίων, καθώς η απλή καλωδίωση της διάυλου και το μεγάλο μήκος καλωδίου είναι ιδανικά για τη σύνδεση απομακρυσμένων συσκευών. Μπορεί να χρησιμοποιηθεί για τον έλεγχο συστημάτων βιντεοεπιτήρησης ή για τη διασύνδεση πάνελ ελέγχου ασφαλείας και συσκευών όπως αναγνώστες καρτών πρόσβασης.

Χρησιμοποιείται επίσης στον Ψηφιακό Έλεγχο Εντολών (DCC) για μοντέλα σιδηροδρόμων. Η εξωτερική διεπαφή στον σταθμό εντολών DCC είναι συχνά RS-485 που χρησιμοποιείται από φορητούς ελεγκτές ή για τον έλεγχο της διαρρύθμισης σε ένα δικτυωμένο περιβάλλον υπολογιστή. Σε αυτή την περίπτωση, χρησιμοποιούνται οι αρθρωτοί συνδετήρες 8P8C.

Μηνύματα

Το RS-485 δεν χρησιμοποιεί κάποιο πρωτόκολλο επικοινωνίας, είναι απλώς μια ηλεκτρική διεπαφή. Αν και πολλές εφαρμογές χρησιμοποιούν τα επίπεδα σημάτων RS-485, η ταχύτητα, η μορφή και το πρωτόκολλο της μετάδοσης δεδομένων δεν καθορίζονται από το RS-485. Η διαλειτουργικότητα ακόμα και παρόμοιων συσκευών από διαφορετικούς κατασκευαστές δεν διασφαλίζεται μόνο με τη συμμόρφωση στα επίπεδα σημάτων.

Σήματα

Signal	Mark (Logic 1)	Space (Logic 0)
A	Low	High
B	High	Low

Η διαφορική γραμμή RS-485 αποτελείται από δύο σήματα:

- A, το οποίο είναι χαμηλό για λογική 1 και υψηλό για λογική 0 και,
- B, το οποίο είναι υψηλό για λογική 1 και χαμηλό για λογική 0.

Επειδή μια κατάσταση mark (λογική 1) παραδοσιακά αναπαρίσταται (π.χ. στο RS-232) με αρνητική τάση και η κατάσταση space (λογική 0) με θετική τάση, το A μπορεί να θεωρηθεί το μη αναστρέψιμο σήμα και το B ως το αναστρέψιμο. Το στανταρ RS-485 αναφέρει (παραφρασμένο):

- Για μια κατάσταση off, mark ή λογική 1, ο ακροδέκτης A του οδηγού είναι αρνητικός σε σχέση με τον ακροδέκτη B.

- Για μια κατάσταση on, space ή λογική 0, ο ακροδέκτης A του οδηγού είναι θετικός σε σχέση με τον ακροδέκτη B.

Οι πίνακες αλήθειας των πιο δημοφιλών συσκευών, ξεκινώντας με το SN75176, δείχνουν τα σήματα εξόδου αντεστραμμένα. Αυτό είναι σύμφωνο με την ονοματολογία A/B που χρησιμοποιούν οι περισσότεροι κατασκευαστές διαφορικών πομποδεκτών, όπως:

- Intersil, όπως φαίνεται στο φύλλο δεδομένων τους για τον πομποδέκτη ISL4489

- Maxim, όπως φαίνεται στο φύλλο δεδομένων τους για τον πομποδέκτη MAX483 και για τον νέο πομποδέκτη μικροελεγκτή 3.3v, τον MAX3485

- Linear Technology, όπως φαίνεται στο φύλλο δεδομένων τους για τους LTC2850, LTC2851, LTC2852

- Analog Devices, όπως φαίνεται στο φύλλο δεδομένων τους για τους ADM3483, ADM3485, ADM3488, ADM3490, ADM3491

- FTDI, όπως φαίνεται στο φύλλο δεδομένων τους για τον USB-RS485-WE-1800-BT

Όλοι αυτοί οι κατασκευαστές συμφωνούν ως προς την έννοια του στανταρ, και η πρακτική τους είναι ευρέως διαδεδομένη. Το ζήτημα επίσης υφίσταται σε εφαρμογές προγραμματιζόμενων λογικών ελεγκτών. Πρέπει να λαμβάνεται προσοχή κατά τη χρήση των ονομάτων A/B. Συχνά χρησιμοποιείται εναλλακτική ονοματολογία για να αποφευχθεί η σύγχυση γύρω από τα ονόματα A/B:

- TX+/RX+ ή D+ ως εναλλακτικό για το B (υψηλό για mark δηλαδή αδράνεια)

- TX−/RX− ή D− ως εναλλακτικό για το A (χαμηλό για mark δηλαδή αδράνεια)

Οι οδηγοί που συμμορφώνονται με το στανταρ RS-485 παρέχουν διαφορική έξοδο τουλάχιστον 1.5 V σε φορτίο 54 Ω, ενώ οι δέκτες που συμμορφώνονται με το στανταρ ανιχνεύουν διαφορική είσοδο έως 200 mV. Οι δύο τιμές παρέχουν επαρκές περιθώριο για αξιόπιστη μετάδοση δεδομένων ακόμη και υπό σοβαρή υποβάθμιση σήματος μέσω καλωδίου και συνδέσμων. Αυτή η ανθεκτικότητα είναι ο κύριος λόγος για τον οποίο το RS-485 είναι κατάλληλο για δικτύωση μεγάλων αποστάσεων σε θορυβώδη περιβάλλοντα.

Εκτός από τις συνδέσεις A και B, μπορεί να υπάρχει μια προαιρετική, τρίτη σύνδεση (το σταντα TIA απαιτεί την παρουσία μιας κοινής διαδρομής επιστροφής μεταξύ όλων των κυκλωμάτων γείωσης κατά μήκος της ισορροπημένης γραμμής για σωστή λειτουργία), που

ονομάζεται SC, G ή αναφορά, η κοινή γείωση αναφοράς σήματος που χρησιμοποιείται από τον δέκτη για να μετρήσει τις τάσεις A και B. Αυτή η σύνδεση μπορεί να χρησιμοποιηθεί για να περιοριστεί το κοινό σήμα που μπορεί να εντυπωθεί στις εισόδους του δέκτη. Η επιτρεπτή τάση κοινής λειτουργίας είναι στην περιοχή από -7 V έως +12 V, δηλαδή ±7 V επιπλέον της τάσης σήματος 0-5 V. Η αποτυχία να παραμείνετε εντός αυτού του εύρους θα έχει ως αποτέλεσμα, στην καλύτερη περίπτωση, παραμόρφωση του σήματος και, στη χειρότερη, ζημιά στις συνδεδεμένες συσκευές.

Πρέπει να λαμβάνεται προσοχή ώστε μια σύνδεση SC, ειδικά σε μακροχρόνιες καλωδιώσεις, να μην επιχειρεί να συνδέσει διαφορετικές γειώσεις μαζί - είναι σοφό να προστίθεται κάποιος περιορισμός ρεύματος στη σύνδεση SC. Οι γειώσεις μεταξύ κτιρίων μπορεί να διαφέρουν κατά μια μικρή τάση, αλλά με πολύ χαμηλή αντίσταση και συνεπώς τη δυνατότητα καταστροφικών ρευμάτων - αρκετά για να λιώσουν τα καλώδια σήματος, τις διαδρομές PCB και τις συσκευές πομποδέκτη.

Το RS-485 δεν καθορίζει κανέναν σύνδεσμο ή pinout. Τα κυκλώματα μπορούν να τερματιστούν σε βιδωτούς ακροδέκτες, υποδοχές D-subminiature ή άλλους τύπους συνδέσμων. Το στανταρ δεν συζητά την θωράκιση καλωδίων, αλλά κάνει κάποιες συστάσεις για προτιμώμενες μεθόδους διασύνδεσης της κοινής αναφοράς σήματος και των γειώσεων του εξοπλισμού.

Παράδειγμα κυματομορφής

Το παρακάτω σχήμα δείχνει τα δυναμικά του A(μπλέ) και του B(κόκκινο) ακροδέκτη μιας γραμμης RS-485 κατα την διάρκεια μιας μετάδοσης ενός byte(0xD3, λιγότερα συμμαντικό bit πρώτα) χρησιμοποιώντας μια ασύγχρονη start-stop μέθοδο.

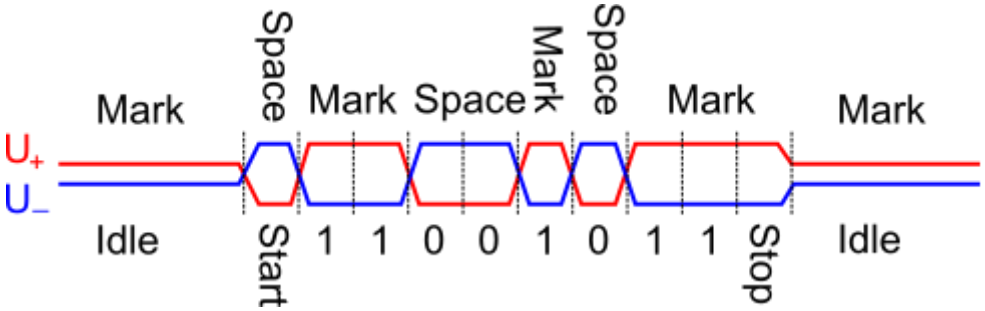


Fig. 3 (B, U+, αναστρεφόμενο) Σήμα σε χρώμα κόκκινο

A(U-, μη-αναστρεφόμενο) Σήμα σε χρώμα μπλέ

(Source: https://en.wikipedia.org/wiki/RS-485)

2.7. Protocol Buffers

Τα Protocol Buffers είναι ένα δωρεάν και ανοιχτού πηγαίου κώδικα φορμάτ δεδομένων που χρησιμοποιείται για την σειριοποίηση δομημένων δεδομένων. Είναι χρήσιμα για την ανάπτυξη προγραμμάτων που επικοινωνούν μεταξύ τους δια μέσω του δικτύου ή για την

αποθήκευση δεδομένων. Η μέθοδος περιλαμβάνει μια γλώσσα περιγραφής διεπαφής η οποία περιγράφει την δομή κάποιων δεδομένων και ένα πρόγραμμα που δημιουργεί κώδικα από την περιγραφή των δεδομένων αυτών για την σειριοποίηση και την αποσειριοποίηση δεδομένων που αντιπροσωπεύουν τα δομημένα δεδομένα.

Τα Protocol Buffers σχεδιάστηκαν από την Google για χρήση μέσα στην εταιρία και αργότερα παρείχαν έναν code generator για χρήση με πολλαπλές γλώσσες χρησιμοποιώντας άδεια ανοιχτού πηγαίου κώδικα. Σχεδιάστηκαν με σκοπό την απλότητα και την αποδοτικότητα, συγκεκριμένα σχεδιάστηκαν για να είναι μικρότερα και πιο γρήγορα από την XML. Η Google τα χρησιμοποιεί για την αποθήκευση και μετάδοση όλων των ειδών πληροφορίες. Τα Protocol Buffers είναι παρόμοια με το Thrift της Apache, το Ion της Amazon και τα Bond Protocols της Microsoft, προσφέροντας ένα ξεκάθαρο protocol stack για RPC (Inter-machine communication procedure call) για την χρήση σε καθορισμένες υπηρεσίες ονόματες gRPC [25].

Δομές δεδομένων (ονόματες "μηνύματα") και υπηρεσίες περιγράφονται σε ένα αρχείο ορισμού proto (.proto) που γίνεται compiled με την χρήση του protoc. Το compilation δημιουργεί κώδικα που μπορεί να κληθεί από έναν αποστολέα ή από έναν παραλήπτη αυτών των δομών δεδομένων. Για παράδειγμα, τα αρχεία example.pb.cc και example.pb.h δημιουργούνται απο το example.proto. Καθορίζουν κλάσεις C++ για κάθε μήνυμα και υπηρεσία στο example.proto. Τα μηνύματα σειριοποιούνται σε δυαδικό φορματ wire το οποίο είναι μικρό σε μέγεθος, έχουν συμβατότητα μπρος και πίσω (forward and backward compatible) αλλά δεν περιγράφονται μόνα τους, αυτό σημαίνει ότι χωρίς το αρχείο .proto δεν υπάρχει τρόπος να ξέρει κάποιος τα ονόματα, την έννοια και τους τύπους των μεταβλητών.

Ενώ ο κύριος σκοπός των Protocol Buffers είναι η χρήση τους για την επικοινωνία σε δίκτυα, η απλότητα και η ταχύτητα τους τα καθιστούν καλά εναλλακτική λύση για κλάσεις και δομές δεδομένων της C++, ειδικά όταν η υποστήριξη για πολλαπλές γλώσσες ή συστήματα είναι ανάγκη για το μέλλον. Τα protobufs δεν έχουν κάποια συγκεκριμένη προδιαγραφή. Το φορματ είναι κατάλληλη για μικρά κομμάτια δεδομένων που δεν ξεπερνούν τα μερικά megabytes και μπορούν φορτωθούν ή σταλούν στην μνήμη αμέσως και επομένως δεν κάνουν για δεδομένα για streaming. Η βιβλιοθήκη δεν προσφέρει συμπίεση και το φορματ δεν έχει καλή υποστήριξη για μη αντικειμενοστραφής γλώσσες προγραμματισμού.

Τα protocol buffers ξεχωρίζουν όσον αφορά την επίδοση και την απόδοση σε σχέση με άλλες τεχνικές σειριοποίησης όπως το JSON, το XML και το Apache Avro. Τα protobufs χρησιμοποιούν binary φορματ για την σειριοποίηση των δεδομένων που σημαίνει ότι τα μηνύματα πιάνουν λίγο χώρο και ότι υπάρχει βελτίωση στην ανάλυση (parsing) του μηνύματος, αυτό τα καθιστά ιδανικά για εφαρμογές και συστήματα που χρήζουν υψηλής απόδοσης. Τα protobufs είναι επίσης χρήσιμα για εφαρμογές μετάδοσης δεδομένων που το bandwidth είναι περιορισμένο. Το JSON σε αντίθεση, ενώ είναι πολύ εύκολο στη χρήση, μπορεί να διαβαστεί από άνθρωπο στην σειριοποιημένη του μορφή παραμένει να είναι κείμενο που σημαίνει ότι πιάνει πολύ χώρο το οποίο οδηγεί σε μεγαλύτερο χρόνο ανάλυσης.

Το XML είναι ακόμα χειρότερο στους ίδιους τομείς διότι ο τρόπος με τον οποίο γράφεται είναι περιττός, προφανώς αυτό οδηγεί σε ακόμα μεγαλύτερα μηνύματα τα οποία πιάνουν ακόμα περισσότερο χώρο. Το Avro λειτουργεί με παρόμοιο τρόπο με τα protobufs με την έννοια ότι χρησιμοποιεί binary φορματ για την σειριοποίηση του και παρέχει παρόμοιες επιδόσεις με τα protobufs. Μια μεγάλη διαφορά ανάμεσα σε όλες αυτές τις τεχνολογίες είναι

ότι τα protobufs και το Avro χρησιμοποιούν schema, ενώ τα JSON και XML δεν έχουν schema. Το schema προσφέρει ακεραιότητα δεδομένων και δακτυλογραφημένο ορισμό των τύπων των μεταβλητών. Τα protobufs χρησιμοποιούν ένα αρχείο .proto το οποίο πρέπει να γίνει compiled πριν μπορέσει να γίνει σειριοποίηση ή αποσειριοποίηση των δεδομένων, το οποίο εξασφαλίζει ότι τα δεδομένα θα έχουν την δομή που θέλουμε και έχουμε θέσει.

Η χρήση του schema μας δίνει την δυνατότητα για forward και backward compatibility και έτσι μας επιτρέπει να αλλάξουμε τη δομή των δεδομένων μας όσο η εφαρμογή μας αλλάζει και αναπτύσσεται, χωρίς όμως να χαλάμε τα ήδη υπάρχοντα συστήματα που έχουμε θέσει σε λειτουργία. Το Avro ενσωματώνει το schema της δομής δεδομένων μέσα στα σειριοποιημένα δεδομένα το οποίο κάνει τα μηνύματα να είναι λίγο μεγαλύτερα από τα αυτά των protobuf για την μετάδοση. Το JSON δεν χρειάζεται schema, το οποίο προσφέρει προσαρμοστικότητα αλλά με κόστος την ακεραιότητα των δεδομένων και τον τύπο των μεταβλητών. Το XML έχει προερετικό schema το οποίο προσφέρει αρκετά γερή δυνατότητα επικύρωσης των δεδομένων αλλά την ίδια στιγμή αυξάνεται πολύ η πολυπλοκότητα χρήσης του.

Όσον αφορά την ευκολία χρήσης το JSON είναι το πρώτο σαν επιλογή και η πιο χρησιμοποιημένη τεχνολογία σειριοποίησης. Είναι αναγνώσιμο από ανθρώπους και δεν χρειάζεται κανένα στήσιμο ή εγκατάσταση, πράγμα που το καθιστά την κατάλληλη επιλογή για χρήση σε web API, σε ρυθμίσεις παραμέτρων, και σενάρια όπου η γρήγορη ενσωμάτωση είναι σημαντική. Τα Protocol Buffers και το Avro σε αντίθεση χρειάζονται τον ορισμό του schema, και την εγκατάσταση κάποιου λογισμικού για να γίνει το compilation του schema και έτσι η παραγωγή του κώδικα αλλά με το θετικό ότι μετά την πρώτη ρύθμιση προσφέρουν πολύ καλύτερο αποτέλεσμα. Κάθε τεχνολογία έχει τα δυνατά και τα αδύναμα στοιχεία της με τα Protocol buffers να έχουν αρκετά καλά γνωρίσματα και από τις δύο πλευρές καθιστώντας τα την κατάλληλη επιλογή για χρήση στα δίκτυα LoRaWAN.

2.8. LoRa και LoRaWAN

Το LoRa (ακρωνύμιο του "long range") είναι μια φυσική ιδιωτική μέθοδος ραδιοεπικοινωνίας. Βασίζεται σε τεχνικές διαμόρφωσης διασποράς φάσματος [16] που προέρχονται από την τεχνολογία διασποράς φάσματος chirp (CSS) [17]. Αναπτύχθηκε από την Cycleo, μια εταιρεία του Γκρενόμπλ, Γαλλία, και κατοχυρώθηκε με δίπλωμα ευρεσιτεχνίας το 2014. Η Cycleo αργότερα αποκτήθηκε από την Semtech.

Το LoRaWAN (wide area network) ορίζει το πρωτόκολλο επικοινωνίας και την αρχιτεκτονική του συστήματος. Το LoRaWAN είναι ένα επίσημο πρότυπο της Διεθνούς Ένωσης Τηλεπικοινωνιών (ITU), ITU-T Y.4480. Η συνεχής ανάπτυξη του πρωτοκόλλου LoRaWAN διαχειρίζεται από την ανοιχτή, μη κερδοσκοπική συμμαχία LoRa Alliance, της οποίας η Semtech είναι ιδρυτικό μέλος.

Το LoRa και το LoRaWAN ορίζουν από κοινού ένα πρωτόκολλο δικτύου χαμηλής κατανάλωσης ευρείας περιοχής (LPWA) που έχει σχεδιαστεί για να συνδέει ασύρματα συσκευές με μπαταρία στο διαδίκτυο σε περιφερειακά, εθνικά ή παγκόσμια δίκτυα και να ανταποκρίνεται σε βασικές απαιτήσεις για το Διαδίκτυο των πραγμάτων (IoT). Παραδείγματα περιλαμβάνουν δύο τρόποι επικοινωνίας, από άκρο σε άκρο ασφάλεια, κινητικότητα και υπηρεσίες τοποθεσίας. Η χαμηλή κατανάλωση ενέργειας, οι χαμηλοί ρυθμοί δεδομένων και η

χρήση στο Internet of Things διακρίνουν αυτόν τον τύπο δικτύου από τα ασύρματα WAN, τα οποία έχουν σχεδιαστεί για να συνδέουν χρήστες ή επιχειρήσεις και να μεταδίδουν περισσότερα δεδομένα, χρησιμοποιώντας περισσότερη ενέργεια. Οι ρυθμοί δεδομένων LoRaWAN κυμαίνονται από 0,3 kbit/s έως 50 kbit/s ανά κανάλι.

Το LoRa χρησιμοποιεί ραδιοσυχνότητες ελεύθερης χρήσης χωρίς άδειες, sub-gigahertz.

- EU868 (863–870/873 MHz) στην Ευρώπη
- AU915/AS923-1 (915–928 MHz) στη Νότια Αμερική
- US915 (902–928 MHz) στη Βόρεια Αμερική
- IN865 (865–867 MHz) στην Ινδία
- AS923 (915–928 MHz) στην Ασία

Το LoRa επιτρέπει εκπομπές μεγάλου εύρους με μικρή κατανάλωση ενέργειας. Αυτή η τεχνολογία καλύπτει το φυσικό επίπεδο, όσο άλλες τεχνολογίες και πρωτόκολλα όπως το LoRaWAN (Δίκτυο ευρείας περιοχής μεγάλης εμβέλειας) καλύπτουν τα ανώτερα επίπεδα. Το LoRaWAN μπορεί να καταφέρει ρυθμούς δεδομένων μεταξύ 0,3 kbit/s και 27 kbit/s, αναλόγως τον παράγοντα διασποράς.

Το LoRa είναι μία από τις πιο δημοφιλείς τεχνολογίες ασύρματων δικτύων αισθητήρων χαμηλής ισχύος για την υλοποίηση του Internet of Things, προσφέροντας επικοινωνία μεγάλου εύρους σε σύγκριση με τεχνολογίες όπως το Zigbee ή το Bluetooth, αλλά με χαμηλότερους ρυθμούς δεδομένων. Οι συσκευές LoRa έχουν ικανότητες γεωεντοπισμού που βλέπουν χρήση στον τριγωνισμό θέσεων των ίδιων των συσκευών μέσω χρονικών σφραγίδων από πύλες(gateways).

LoRa PHY

Το LoRa χρησιμοποιεί ιδιόκτητη διαμόρφωσης διασποράς φάσματος που είναι παρόμοια και καρπός της τεχνολογίας διασποράς φάσματος chirp (CSS). Κάθε σύμβολο αντιπροσωπεύεται από ένα κυκλικά μετατοπισμένο chirp στο συχνοτικό διάστημα ($f_0 - B/2, f_0 + B/2$) όπου $f0$ είναι το κέντρο της συχνότητας και B το εύρος ζώνης(bandwidth) του σήματος σε Hertz. Ο παράγοντας διασποράς (spread factor, SF) είναι μια επιλεγόμενη παράμετρος ράδιο από 5 έως 12 [18] και αντιπροσωπεύει τον αριθμό των bits που αποστέλλονται ανά σύμβολο και επιπλέον καθορίζει πόσο διασκορπίζεται η πληροφορία στον χρόνο [17].

Υπάρχουν $M = 2^{SF}$ διαφορετικές αρχικές συχνότητες του κυκλικά μετατοπισμένου chirp (η στιγμιαία συχνότητα αυξάνεται γραμμικά και τυλίγεται στο $f_0 - B/2$ όταν φτάνει την μέγιστη συχνότητα $f_0 + B/2$) [19]. Ο ρυθμός του συμβόλου καθορίζεται απο $R_s = B/2^{SF}$. Το LoRa έχει την δυνατότητα να ανταλλάξει ταχύτητα μετάδοσης δεδομένων για ευαισθησία (δεδομένου ότι το εύρος ζώνης B του καναλιού είναι σταθερό) επιλέγοντας την διασπορά φάσματος (SF). Χαμηλότερη διασπορά φάσματος σημαίνει ότι υπάρχει μεγαλύτερος ρυθμός δεδομένων αλλά χειρότερη ευαισθησία ενώ υψηλότερη διασπορά φάσματος έχει καλύτερη ευαισθησία και μικρότερο ρυθμό δεδομένων.

Συγκρίνοντας την μικρή διασπορά φάσματος, για να στείλουμε την ίδια ποσότητα δεδομένων με μεγάλη διασπορά φάσματος χρειάζεται περισσότερος χρόνος μετάδοσης, γνωστός και ως time-on-air(χρόνος στον αέρα). Όσο μεγαλύτερο time on air έχει ένα μήνυμα τόσο παραπάνω χρόνο λειτουργεί το μοντεμ έτσι καταναλώνοντας περισσότερη ενέργεια. Συνήθως τα LoRa μόντεμ υποστηρίζουν ισχύ μετάδοσης μέχρι και +22dBm [18]. Όμως, οι κανονισμοί αντίστοιχης χώρας μπορεί να περιορίζουν την ισχύ μετάδοσης παραπάνω. Η μεγαλύτερη ισχύ μετάδοσης οδηγεί σε μεγαλύτερη ισχύ σήματος στο δέκτη και επομένως μεγαλύτερος προϋπολογισμός σύνδεσης (link budget) [20] με κόστος μεγαλύτερη κατανάλωση ενέργειας.

Υπάρχουν μελέτες μέτρησης της απόδοσης του LoRa όσον αφορά την κατανάλωση ενέργειας, τις αποστάσεις επικοινωνίας και την αποτελεσματικότητα πρόσβασης στο μέσο [21]. Σύμφωνα με την Πύλη Ανάπτυξης του LoRa, η εμβέλεια που παρέχεται από το LoRa μπορεί να φτάσει έως και 4,8 χιλιόμετρα σε αστικές περιοχές και έως και 16 χιλιόμετρα ή περισσότερο σε αγροτικές περιοχές (οπτική επαφή) [22]. Επιπλέον, το LoRa χρησιμοποιεί κωδικοποίηση προώθησης σφαλμάτων για να βελτιώσει την ανθεκτικότητα έναντι παρεμβολών. Η υψηλή εμβέλεια του LoRa χαρακτηρίζεται από υψηλά προϋπολογίσματα ασύρματων συνδέσεων περίπου από 155 dB έως 170 dB.

LoRaWAN

Το LoRa αφορά το χαμηλότερο, φυσικό, επίπεδο, και τα ανώτερα επίπεδα δικτύου αναλαμβάνονται από το LoRaWAN. Το LoRaWAN είναι ένα πρωτόκολλο που αναπτύχθηκε για να ορίσει τα ανώτερα επίπεδα του δικτύου. Το LoRaWAN είναι ένα πρωτόκολλο ελέγχου πρόσβασης στο μέσο (MAC) βασισμένο στο cloud, αλλά λειτουργεί κυρίως ως πρωτόκολλο επιπέδου δικτύου για τη διαχείριση της επικοινωνίας μεταξύ πυλών LPWAN και συσκευών τερματικού κόμβου(end-node), ως ένα πρωτόκολλο δρομολόγησης που διατηρείται από την LoRa Alliance.

Το LoRaWAN καταρτίζει το πρωτόκολλο επικοινωνίας και την αρχιτεκτονική συστήματος για το δίκτυο, ενώ το φυσικό επίπεδο του LoRa δίνει την δυνατότητα για μακρά επικοινωνία. Το LoRaWAN διαχειρίζεται επίσης τη συχνότητα επικοινωνίας, τον ρυθμό δεδομένων και την ισχύ όλων των συσκευών. Οι συσκευές στο δίκτυο είναι ασύγχρονες και μεταδίδονται όταν είναι διαθέσιμα δεδομένα για αποστολή. Τα δεδομένα που αποστέλλονται από συσκευές τερματικού κόμβου λαμβάνονται από πολλαπλές πύλες που προωθούν τα πακέτα σε έναν κεντρικό διακομιστή δικτύου. Τα δεδομένα στη συνέχεια προωθούνται σε διακομιστές εφαρμογών. Αυτή η τεχνολογία εμφανίζει υψηλή αξιοπιστία για μέτρια φορτία, ωστόσο, έχει μερικά προβλήματα απόδοσης με την αποστολή αναγνωρίσεων.

LoRaWAN Duty Cycle

Το duty cycle (κύκλος λειτουργίας) αναφέρεται στην ρύθμιση του χρόνου κατα τον οποίο δίνεται η δυνατότητα σε μια LoRaWAN συσκευή να μεταδώσει για έναν συγκεκριμένο χρόνο. Είναι ένα μέτρο για την εξασφάλιση της δίκαιης χρήσης του διαθέσιμου φάσματος και για την μείωση των παρεμβολών με άλλες συσκευές. Το duty cycle συνήθως εκφράζεται ως

ποσοστό, το οποίο υποδεικνύει την αναλογία του χρόνου που μια συσκευή μπορεί να μεταδίδει ενεργά εναντίον του συνολικού διαθέσιμου χρόνου.

Ο ορισμός του duty cycle υπολογίζεται έτσι: $Duty\ Cycle = (\frac{Transmission\ Time}{Total\ Time}) \times 100$

Διάφορες χώρες έχουν ρυθμιστικές αρχές οι οποίες επιβάλλουν περιορισμούς στο duty cycle για να εξασφαλιστεί η αποδοτική και δίκαιη χρήση της ραδιοφωνικής συχνότητας. Για παράδειγμα στην Ευρωπαϊκή Ένωση, το duty cycle για τις συσκευές LoRaWAN κυβερνάται από το ETSI (European Telecommunications Standards Institute) κανονισμούς.

Αν σε μια συσκευή επιτρέπεται ένα duty cycle του 1%, σημαίνει ότι η συσκευή μπορεί να μεταδώσει για το 1% του συνολικού χρόνου. Σε μια ώρα (3600 δευτερόλεπτα), σημαίνει :

$$Transmission\ Time\ = 0.01\ \times\ 3600\ =\ 36\ seconds$$

Άρα αυτή η συσκευή μπορεί να μεταδώσει μόνο για 36 δευτερόλεπτα κάθε ώρα.

Οι συσκευές πρέπει να προσέχουν και να διαχειρίζονται σωστά τους χρόνους μετάδοσης τους. Αν μια συσκευή ξεπεράσει τον χρόνο του duty cycle της, πρέπει να περιμένει μέχρι να γίνει επαναφορά της περιόδου του duty cycle πριν μπορέσει να μεταδώσει ξανά. Οι περιορισμοί αυτοί αφορούν τόσο τα uplinks όσο και τα downlinks. Η ισορροπία είναι κρίσιμη για την διατήρηση της απόδοσης του δικτύου. Η σωστή εφαρμογή αυτών των περιορισμών μπορεί να οδηγήσει σε μικρότερη κατανάλωση ενέργειας το οποίο είναι πολύ ωφέλιμο σε περίπτωση που η LoRaWAN συσκευή λειτουργεί με μπαταρία. Ένα ακόμα πλεονέκτημα του duty cycle είναι ότι αποτρέπει την συμφόρηση του δικτύου περιορίζοντας τον χρόνο που κάθε μια συσκευή μπορεί να διαμένει στο κανάλι, έτσι επιτρέποντας παραπάνω συσκευές να μοιράζονται την ίδια συχνότητα.

Σε περιοχές με πιο αυστηρούς κανονισμούς γίνεται χρήση άλλων μηχανισμών όπως το Adaptive Data Rate (ADR) και το channel hopping για την βελτιστοποίηση του δικτύου όσο τηρούνται οι περιορισμοί του duty cycle.

CSMA για το LoRaWAN

Στις ασύρματες επικοινωνίες, ειδικά στον τομέα του Διαδικτύου των Πραγμάτων (IoT), η αποτελεσματική χρήση του καναλιού και η αποφυγή συγκρούσεων είναι ουσιώδους σημασίας για την αξιοπιστία του δικτύου και την φασματική απόδοση. Προηγουμένως, το LoRaWAN βασιζόταν στο ALOHA ως πρωτόκολλο ελέγχου πρόσβασης στο μέσο (MAC), αλλά για να βελτιωθεί αυτό, η Τεχνική Σύσταση TR013 της LoRa Alliance εισάγει το CSMA-CA, το οποίο προσαρμόζεται για να λαμβάνει υπόψη τα χαρακτηριστικά της διαμόρφωσης του LoRa, συμπεριλαμβανομένης της ορθογωνικότητας του Συντελεστή Διάδοσης (Spreading Factor) και της δυνατότητας επικοινωνίας κάτω από το επίπεδο θορύβου.

Αυτή η προσαρμογή ξεπερνά τους περιορισμούς της απλής ανίχνευσης βάσει της ισχύος του λαμβανόμενου σήματος (RSS), η οποία μπορεί να διαταράξει αυτές τις μοναδικές ιδιότητες. Η εφαρμογή της TR013 βελτιώνει τη φασματική απόδοση του LoRaWAN και εξασφαλίζει πιο αξιόπιστη επικοινωνία με τις συσκευές, συμπεριλαμβανομένων και των περιβαλλοντικά φορτισμένων περιβαλλόντων.

Listen Before Talk(LBT)

Το Listen Before Talk είναι μια τεχνολογία διαχείρισης φάσματος που χρησιμοποιείται σε δίκτυα LoRaWAN για να εξασφαλίσει ότι οι συσκευές δεν προκαλούν παρεμβολές μεταξύ τους όταν εκπέμπουν. Η τεχνολογία LBT είναι ιδιαίτερα σημαντική σε περιβάλλοντα όπου πολλές συσκευές μοιράζονται το ίδιο φάσμα και σε περιοχές όπου οι κανονισμοί απαιτούν τη χρήση της για την διαχείριση της χρήσης φάσματος, όπως στη Νότια Κορέα.

Μια τερματική συσκευή ή πύλη με LBT πριν ξεκινήσει να μεταδίδει δεδομένα, “ακούει” το κανάλι στο οποίο θέλει να μεταδώσει για κάποιο χρονικό διάστημα για να βεβαιωθεί ότι είναι ελεύθερο. Αν το κανάλι είναι κατειλημμένο από άλλη μετάδοση, η συσκευή θα παραμείνει έως ότου το κανάλι να γίνει διαθέσιμο. Αν το κανάλι είναι ελεύθερο, η συσκευή μπορεί να προχωρήσει στην μετάδοση των δεδομένων της. Αν το κανάλι είναι συνεχώς κατειλημμένο, η συσκευή θα αναλάβει τη διαδικασία ακρόασης μετά από έναν προκαθορισμένο χρόνο.

Το LBT έχει αρκετά πλεονεκτήματα για τα LoRaWAN δίκτυα, όπως την μείωση των παρεμβολών, μειώνει τις πιθανότητες παρεμβολών μεταξύ των συσκευών που εκπέμπουν στο ίδιο φάσμα, βελτιώνοντας έτσι τη συνολική αξιοπιστία του δικτύου. Ένα άλλο πλεονέκτημα είναι ότι γίνεται πιο αποτελεσματική χρήση του φάσματος, το φάσμα χρησιμοποιείται πιο αποδοτικά, καθώς οι συσκευές περιμένουν να γίνει διαθέσιμο το κανάλι πριν μεταδώσουν, αποφεύγοντας έτσι την σπατάλη πόρων.

Αρχιτεκτονική LoRaWAN

Τα δίκτυα LoRaWAN χρησιμοποιούνται σε τοπολογίες αστέρας αστέρων και αποτελείται από τα παρακάτω στοιχεία.

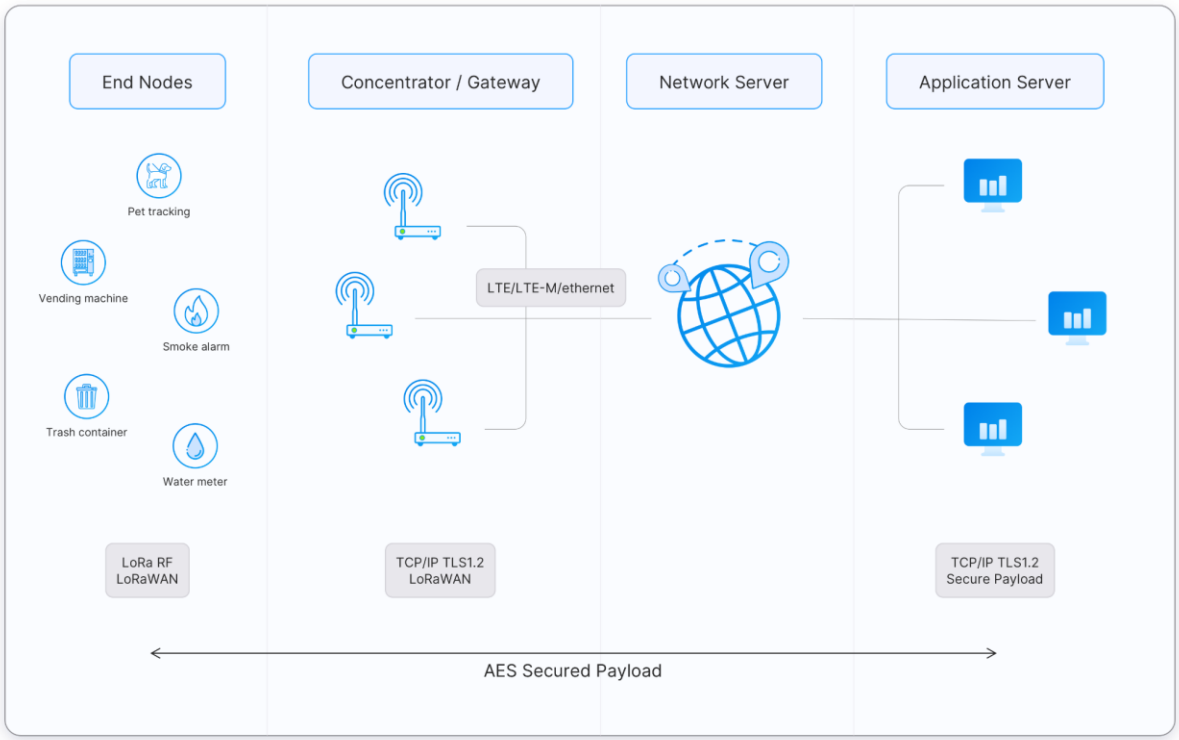


Fig 4. Typical LoRaWAN Network Architecture (source: https://www.thethingsnetwork.org/docs/lorawan/architecture/)

- **Τερματική συσκευή** - Μπορούν να είναι αισθητήρες ή ενεργοποιητές τα οποία στέλνουν ή δέχονται LoRa διαμορφωμένα μηνύματα πρός και από τις πύλες LoRa (LoRa Gateways)
- **Πύλη LoRa (Gateway)** - Δέχονται μηνύματα από τις τερματικές συσκευές και τα στέλνουν στον Network Server
- **Network Server** - Είναι ένα κομμάτι λογισμικού που τρέχει σε ένα μηχάνημα server που διαχειρίζεται ολόκληρο το δίκτυο.
- **Application Servers** - Είναι ένα κομμάτι λογισμικού που επίσης τρέχει σε μηχάνημα server και είναι υπεύθυνο την ασφαλή επεξεργασία των δεδομένων την εφαρμογής
- **Join Server** - Είναι ένα κομμάτι λογισμικού που τρέχει σε μηχάνημα server και επεξεργάζεται τα μηνύματα αίτησης εγγραφής των τερματικών συσκευών, το Join Server δεν εμφανίζεται στο πάνω σχέδιο.

Οι τερματικές συσκευές επικοινωνούν με τις πύλες που βρίσκονται στην εμβέλεια τους και κάθε πύλη συνδέεται με το network server. Τα δίκτυα LoRaWAN χρησιμοποιούν πρωτόκολλα βασισμένα στο ALOHA οπότε οι συσκευές δεν χρειάζεται να επικοινωνούν με συγκεκριμένες πύλες. Τα μηνύματα που στέλνονται από τις τερματικές συσκευές μεταφέρονται μέσα από όλες τις πύλες που βρίσκονται σε εμβέλεια. Τα μηνύματα λαμβάνονται απο το Network Server, αν το Network Server δεχτεί πολλαπλά του ίδιου μηνύματος κρατάει μόνο ένα και αγνοεί τα υπόλοιπα αντίγραφα. Αυτό λέγεται message deduplication.

Ροή επικοινωνίας LoRaWAN

- Uplink - Η τερματική συσκευή στέλνει μήνυμα πρός το gateway
- Gateway - Το gateway δέχεται το uplink και το στέλνει στο Network server
- Το Network server επεξεργάζεται τα δεδομένα, τους εφαρμόζει οποιαδήποτε πρωτόκολλα χρειάζεται και τα στέλνει στο application servers
- Downlink - Αν χρειαστεί το network server στέλνει κάποια εντολή ή δεδομένα πίσω στη τερματική συσκευή μέσω του gateway.

End Device

Στο δίκτυο LoRaWAN μια τερματική συσκευή είναι μια ασύρματη συσκευή η οποία χρησιμοποιεί LoRa modulation για να επικοινωνήσει με gateways και διαμέσου τους με το network server. Είναι συνήθως συσκευές με αισθητήρες και ή ενεργοποιητές και χρησιμοποιούνται σε διάφορες εφαρμογές IoT. Τα κύρια χαρακτηριστικά τους είναι:

- Επικοινωνία - επικοινωνούν με διάφορα gateways στέλνοντας Uplinks και δέχονται downlinks.
- Ενεργειακή αποδοτικότητα - οι συσκευές αυτές συνήθως έχουν ως παροχή ενέργειας μπαταρίες και έχουν πολύ καλή επίδοση το οποίο τις επιτρέπει να λειτουργούν για μεγάλα χρονικά διαστήματα χωρίς την αλλαγή της μπαταρίας.
- Κλάση συσκευής - Υπάρχουν οι κλάσεις A,B και C. Η συσκευές κλάσης A είναι οι πιό ενεργειακά αποδοτικές, ανοίγουν ένα παράθυρο λήψης (receive window) μόνο αφού στείλουν μήνυμα, το οποίο τα καθιστά κατάλληλα για εφαρμογές που χρειάζονται πολύ λίγα downlinks. Οι συσκευές κλάσης B έχουν προκαθορισμένα παράθυρα λήψης το οποίο τους επιτρέπει να έχουν πιο προβλέψιμα downlinks. Οι συσκευές κλάσης C αφήνουν το παράθυρο λήψης σχεδόν πάντα έτσι επιτρέποντας downlinks μικρής καθυστέρησης αλλά με μεγαλύτερο ενεργειακό κόστος.
- Κινητικότητα - Οι συσκευές μπορούν να είναι είτε σταθερές είτε κινητές αναλόγως τις απαιτήσεις της εφαρμογής.

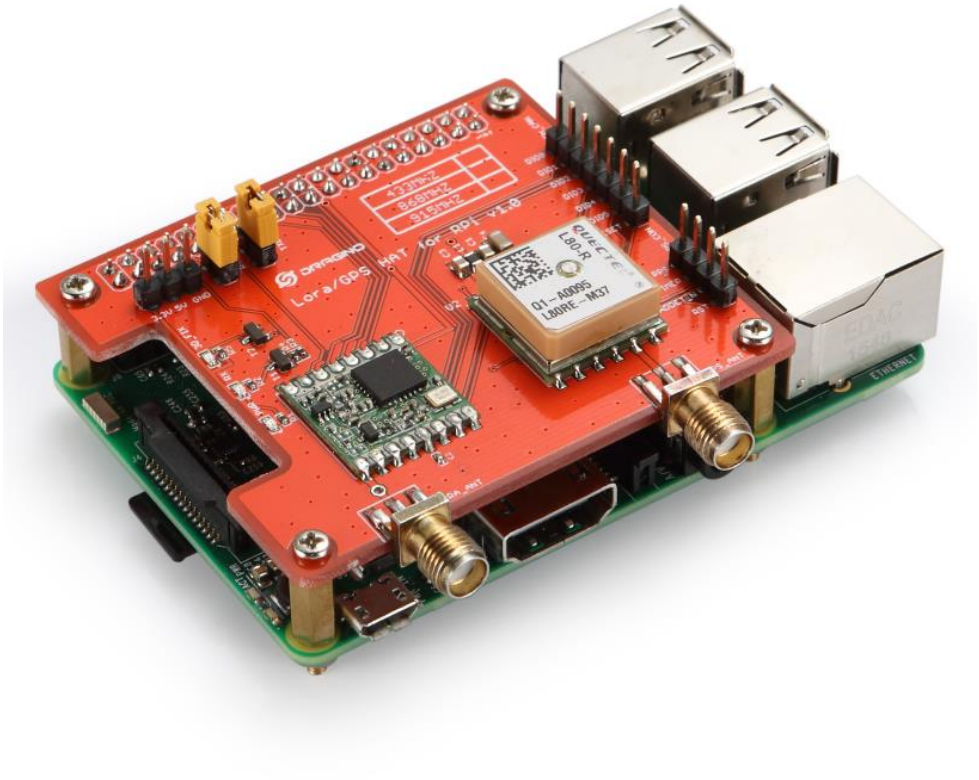


Fig. 5 End Device using a Raspberry Pi 3 and a Dragino Pi Hat (source: https://www.dragino.com/products/lora/item/106-lora-gps-hat.html)

Gateway

Σε ένα δίκτυο LoRaWAN, ένα Gateway είναι ένα κρίσιμο εξάρτημα στην υποδομή για να λαμβάνει και να μεταδίδει ραδιοφωνικά σήματα LoRa μεταξύ των end devices και και του network server. Τα gateways εξυπηρετούν ως μεσάζοντας μεταξύ των τερματικών συσκευών και της δικτυακής υποδομής και διοργανώνουν την αμφίδρομη επικοινωνία μεγάλων αποστάσεων. Τα κύρια χαρακτηριστικά είναι:

- Διεπαφή Radio - Τα gateways είναι εξοπλισμένα με πομποδέκτες LoRa Radio οι οποίοι τα επιτρέπουν να επικοινωνήσουν με τερματικές συσκευές εντός εμβέλειας χρησιμοποιώντας LoRa modulation.
- Συνδεσιμότητα - Συνήθως έχουν ενσύρματη ή ασύρματη συνδεσιμότητα για backhaul επικοινωνία με το network server. Αυτό μπορεί να είναι, ethernet, wifi, κινητό δίκτυο ή δορυφορική σύνδεση αναλόγως με τις ανάγκες της εφαρμογής.
- Κάλυψη - Προσφέρουν κάλυψη σε μία συγκεκριμένη γεωγραφική περιοχή της οποίας το μέγεθος επηρεάζεται από διάφορους συντελεστές, όπως το ύψος της

κεραίας, την ισχύ μετάδοσης και το έδαφος. Πολλαπλα gateways μπορούν να χρησιμοποιηθούν για να εξασφαλιστεί επαρκής κάλυψη σε μία περιοχή.

- Επεκτασιμότητα - Τα δίκτυα LoRaWAN μπορούν να επεκταθούν προσθέτοντας επιπλέον gateways το οποίο επιτρέπει και στην σύνδεση παραπάνω τερματικών συσκευών.
- Ενσωμάτωση - Χρησιμοποιούν στανταρ πρωτόκολλα επικοινωνίας όπως UDP και MQTT για να ανταλλάξουν εντολές και μηνύματα με το network server.
- Backhaul encryption - Συνήθως υποστηρίζουν κρυπτογράφηση και πρωτόκολλα ασφαλής επικοινωνίας για backhaul συνδέσεις για να εξασφαλίσουν την ακεραιότητα ιδιωτικότητα των δεδομένων.



Fig. 6 The Things Network Kickstarter Gateway (source: https://www.thethingsindustries.com/docs/gateways/models/thethingskickstartergateway/)

Network Server

Σε ένα δίκτυο LoRaWAN, ο Network Server είναι ένα κρίσιμο στοιχείο που είναι υπεύθυνο για τη διαχείριση του δικτύου, τη διασφάλιση ασφαλούς επικοινωνίας και την επεξεργασία δεδομένων που ανταλλάσσονται μεταξύ των τερματικών συσκευών και των διακομιστών εφαρμογών. Ο διακομιστής δικτύου παίζει κεντρικό ρόλο στον συντονισμό της λειτουργίας του δικτύου, αναλαμβάνοντας εργασίες όπως η πιστοποίηση των συσκευών, η δρομολόγηση δεδομένων και η βελτιστοποίηση του δικτύου.

Μερικά από τα χαρακτηριστικά του είναι:

- Κεντρική διαχείριση - Παρέχει κεντρικό έλεγχο και διαχείριση ολόκληρου του δικτύου LoRaWAN.
- Επεκτασιμότητα - Είναι σχεδιασμένος να διαχειρίζεται μεγάλο αριθμό τερματικών συσκευών και πυλών, επεκτείνεται ανάλογα με τις απαιτήσεις του δικτύου.
- Ασφάλεια - Εφαρμόζει ισχυρά πρωτόκολλα ασφαλείας για την πιστοποίηση συσκευών, την κρυπτογράφηση δεδομένων και την διασφάλιση της ακεραιότητας και της εμπιστευτικότητας των επικοινωνιών.

Βασικές λειτουργίες του:

- Πιστοποίηση συσκευών
    - Διασφαλίζει ότι μόνο εξουσιοδοτημένες τερματικές συσκευές μπορούν να συνδεθούν και να επικοινωνήσουν εντός του δικτύου.
    - Χρησιμοποιεί μοναδικά αναγνωριστικά συσκευών (DevEUI) και κλειδιά εφαρμογής (AppKey) για την πιστοποίηση των συσκευών κατα την διαδικασία σύνδεσης.
- Δρομολόγηση δεδομένων
    - Δρομολογεί uplink μηνύματα (δεδομένα από τερματικές συσκευές προς τον application server) και downlink μηνύματα (εντολές ή δεδομένα από τον application server προς τις τερματικές συσκευές) με αποδοτικό τρόπο.
    - Διαχειρίζεται την ροή των δεδομένων μεταξύ των πυλών και των application servers.
- Διαχείριση ADR (Adaptive Data Rate)
    - Βελτιστοποιεί την απόδοση του δικτύου με δυναμική προσαρμογή του ρυθμού δεδομένων και της ισχύος εκπομπής των τερματικών συσκευών.
    - Βοηθά στην εξισορρόπηση του φόρτου δικτύου και στη μεγιστοποίηση της διάρκειας ζωής της μπαταρίας των τερματικών συσκευών.
- Ανίχνευση διπλών μηνυμάτων

    - Αναγνωρίζει και φιλτράρει τα διπλότυπα μηνύματα που λαμβάνονται από πολλαπλές πύλες για να διασφαλίσει την ακεραιότητα των δεδομένων και να αποφύγει την επανάληψη.
- Βελτιστοποίηση Δικτύου
    - Διαχειρίζεται τους πόρους του δικτύου και βελτιστοποιεί την απόδοση του.
    - Ρυθμίζει παραμέτρους όπως το Spread Factor και τα επίπεδα ισχύος για την μείωση των παρεμβολών.
- Ασφάλεια και κρυπτογράφηση
    - Κρυπτογραφεί τα δεδομένα για να εξασφαλίσει ασφαλή μετάδοση δεδομένων
    - Διαχειρίζεται κλειδιά session και αλγόριθμους κρυπτογράφησης για την προστασία την ακεραιότητας και της εμπιστευτικότητας των δεδομένων.
- Διαχείριση Συσκευών
    - Διατηρεί μια βάση δεδομένων με όλες τις καταχωρημένες τερματικές συσκευές, μαζί και των ρυθμίσεων τους αλλά και της κατάστασης τους
    - Διευκολύνει τις ενημερώσεις firmware over the air και τις απομακρυσμένες ρυθμίσεις.
- Καταγραφή και Παρακολούθηση
    - Καταγράφει τη δραστηριότητα του δικτύου και παρέχει σε πραγματικό χρόνο παρακολούθηση της απόδοσης και της υγείας του δικτύου.
    - Δημιουργεί αναφορές και ειδοποιήσεις για τους διαχειριστές του δικτύου.

Αλληλεπίδραση με άλλα στοιχεία

- Πύλες
    - Λαμβάνει πακέτα δεδομένων από πολλαπλές πύλες και τα επεξεργάζεται.
    - Στέλνει μηνύματα καθόδου στις τερματικές συσκευές μέσω των κατάλληλων πυλών.
- Application Server
    - Προωθεί επεξεργασμένα δεδομένα στον Application Server, όπου μπορούν να χρησιμοποιηθούν για διάφορες εφαρμογές IoT.
    - Λαμβάνει εντολές ή δεδομένα από τον Application Server για να τα στείλει στις τερματικές συσκευές.

Application Server

Σε ένα δίκτυο LoRaWAN, ο διακομιστής εφαρμογών (Application Server) είναι ένα ακόμα κρίσιμο στοιχείο που είναι υπεύθυνο για τη διαχείριση των δεδομένων και της λειτουργικότητας της εφαρμογής. Αλληλεπιδρά με τον Network Server και λαμβάνει δεδομένα από τις τερματικές συσκευές μέσω του Network Server.

Βασικές λειτουργίες:

- Επεξεργασία και αποθήκευση δεδομένων
    - Λήψη δεδομένων - Λαμβάνει δεδομένα από uplinks των end devices (μετρήσεις αισθητήρων, ενημερώσεις κατάστασης κ.λ.π.) απο τον Network Server.
    - Ανάλυση δεδομένων - Επεξεργάζεται και αναλύει τα ληφθέντα δεδομένα για την εξαγωγή χρήσιμων πληροφοριών. Αυτό μπορεί να περιλαμβάνει την συγκέντρωση δεδομένων, την εκτέλεση στατιστικής ανάλυσης κ.λ.π.
    - Αποθήκευση δεδομένων - Αποθηκεύει τα επεξεργασμένα δεδομένα σε βάσεις για ιστορική ανάλυση, και μελλοντική αναφορά.
- Λογική εφαρμογής
    - Εφαρμόζει την επιχειρηματική λογική που απαιτείται για συγκεκριμένες εφαρμογές IoT. Για παράδειγμα, σε μια εφαρμογή έξυπνης γεωργίας, ο Application Server μπορεί να αναλύει δεδομένα υγρασίας εδάφους για να καθορίσει πότε να ενεργοποιήσει τα συστήματα άρδευσης.
    - Εκτελεί αυτοματοποιημένες ενέργειες βάσει προκαθορισμένων κανόνων ή συνθηκών. Για παράδειγμα, αν ένας αισθητήρας θερμοκρασίας αναφέρει τιμή πάνω από ένα ορισμένο όριο, ο Application Server μπορεί να ενεργοποιήσει μια ειδοποίηση ή να ενεργοποιήσει ένα σύστημα ψύξης.
- Διοίκηση και έλεγχος
    - Επικοινωνία downlink - Στέλνει εντολές, ενημερώσεις διαμόρφωσης ή μηνύματα ελέγχου στις τερματικές συσκευές μέσω του network server. Αυτό επιτρέπει την απομακρυσμένη διαχείριση και τον έλεγχο των συσκευών.
    - Διαχείριση συσκευών - Υποστηρίζει ενημερώσεις firmware over the air, αλλαγές διαμόρφωσης και παρακολούθηση κατάστασης συσκευών.
- Διεπαφή χρήστη και αλληλεπίδραση
    - Παρέχει πίνακες ελέγχου, αναφορές και εργαλεία οπτικοποίησης για τους χρήστες να παρακολουθούν και να διαχειρίζονται τις εφαρμογές IoT τους.
    - Επιτρέπει στους χρήστες να διαμορφώνουν και να ελέγχουν συσκευές, να ορίζουν όρια για ειδοποιήσεις και να βλέπουν δεδομένα σε πραγματικό χρόνο.
- Ενσωμάτωση με εξωτερικά συστήματα
    - Διασυνδέει με άλλα συστήματα πληροφορικής, υπηρεσίες cloud και εφαρμογές τρίτων. Αυτό μπορεί να περιλαμβάνει την ενσωμάτωση με συστήματα

διαχείρισης επιχειρηματικών πόρων (ERP), συστήματα διαχείρισης πελατειακών σχέσεων (CRM) ή άλλες πλατφόρμες ανάλυσης δεδομένων.

- Υποστηρίζει APIs για προσαρμοσμένες ενσωματώσεις και διαμοιρασμό δεδομένων.

- Αλληλεπίδραση με άλλα στοιχεία
  - Τερματικες συσκευές - Επικοινωνεί έμμεσα με τις τερματικές συσκευές μέσω του network server, στέλνοντας εντολές και λαμβάνοντας δεδομένα.
  - Network server - Δρά ως γέφυρα μεταξύ του application server και των τερματικών συσκευών, δρομολογώντας δεδομένα και εντολές μεταξύ τους. Επίσης διασφαλίζει ότι τα δεδομένα αποκρυπτογραφούνται, πιστοποιούνται και δρομολογούνται σωστά στον κατάλληλο application server.
  - Πύλες - Ο application server δεν αλληλεπιδρά άμεσα με τις πύλες.

Join Server

Σε ένα δίκτυο LoRaWAN, ο join server παίζει κρίσιμο ρόλο στη διαδικασία προσθήκης νέων συσκευών στο δίκτυο, ειδικά κατά τη διαδικασία ενεργοποίησης που είναι γνωστή ως Over The Air Activation.

Βασικές λειτουργίες:

- Αυθεντικοποίηση συσκευών
  - Διαχείριση αιτημάτων σύνδεσης - Ο join server λαμβάνει αιτήματα σύνδεσης από τις τερματικές συσκευές. Αυτά τα αιτήματα περιέχουν το μοναδικό αναγνωριστικό της συσκευής (DevEUI) και το nonce σύνδεσης (DevNonce).
  - Αυθεντικοποίηση - Ο join server αυθεντικοποιεί τη συσκευή χρησιμοποιώντας τα μοναδικά πιστοποιητικά της συσκευής, που περιλαμβάνουν το AppKey το οποίο έχει προ διαμοιραστεί μεταξύ της συσκευής και του join server.
- Δημιουργία κλειδιών συνεδρίας
  - Μόλις η συσκευή αυθεντικοποιηθεί, ο join server δημιουργεί κλειδιά συνεδρίας, τα οποία χρησιμοποιούνται για την κρυπτογράφηση και τη ασφάλεια της επικοινωνίας μεταξύ της συσκευής και του δικτύου.
  - AppSKey και NwkSKey - Ο join server παράγει το App Session Key(AppSKey) και το Network Session Key (NwkSKey) χρησιμοποιώντας το AppKey και το request nonce. Αυτά τα κλειδιά είναι απαραίτητα για την ασφαλή μετάδοση δεδομένων.
- Διανομή κλειδιών συνεδρίας
  - Network Server - Ο join server διαβιβάζει με ασφάλεια το NwkSKey στον network server. Αυτό το κλειδί χρησιμοποιείται από τον network server για να

διασφαλίσει την ακεραιότητα και την αυθεντικότητα των μηνυμάτων που ανταλλάσσονται μεταξύ της τερματικής συσκευής και του δικτύου.

    - Application Server - Ο join server διαβιβάζει με ασφάλεια το AppSKey στον application server. Αυτό το κλειδί χρησιμοποιείται από τον application server για την αποκρυπτογράφηση του περιεχομένου των μηνυμάτων που αποστέλλονται από την τερματική συσκευή.

- Διαχείριση εγγράφων συσκευών
    - Βάση δεδομένων συσκευών - Διατηρεί μια βάση δεδομένων όλων των εγγεγραμμένων συσκευών, των πιστοποιητικών τους και της κατάστασης τους. Αυτό περιλαμβάνει πληροφορίες όπως το DevEUI, το AppKey και άλλα μεταδεδομένα που είναι απαραίτητα για αυθεντικοποίηση και τη διαχείριση των συσκευών.
- Ασφάλεια και συμβατότητα
    - Ασφαλής επικοινωνία - Διασφαλίζει ότι όλες οι επικοινωνίες μεταξύ του join server, του network server και του application server είναι ασφαλείς και κρυπτογραφημένες.
    - Συμμόρφωση με τα πρότυπα - Συμμορφώνεται με τα πρότυπα του πρωτοκόλλου LoRaWAN και τις βέλτιστες πρακτικές ασφαλείας για τη διασφάλιση ενός ισχυρου και ασφαλούς περιβάλλοντος δικτύου.
- Αλληλεπίδραση με άλλα στοιχεία
    - Τερματικές συσκευές - Λαμβάνει join requests απο τις τερματικές συσκευές που προσπαθούν να μπουν στο δίκτυο χρησιμοποιώντας OTAA. Έπειτα δημιουργεί και μοιράζει με ασφαλή τρόπο session keys στις τερματικές συσκευές με το που γίνει επιτυχής επιβεβαίωση.
    - Network Server - Συνεργάζεται με τον network server για να πιστοποιήσει τις συσκευές και να διανέμει τα NwkSKey. Επίσης εξασφαλίζει ότι ο network server έχει τα απαραίτητα κλειδιά για να επικυρώσει και να στείλει μηνύματα στις τερματικές συσκευές.
    - Application Server - Διανέμει τα AppSKey στο application server για να επιτρέψει της αποκρυπτογράφηση των payloads του application. Εξασφαλίζει ότι το application server μπορεί να επικοινωνήσει με ασφάλεια με τις τερματικές συσκευές.

Περιορισμοί Των δικτύων LoRaWAN

Τα δίκτυα LoRaWAN, παρά τα πλεονεκτήματα τους για πολλές εφαρμογές IoT, λόγω της μεγάλης εμβέλειας και της χαμηλής κατανάλωσης ενέργειας, έχουν ορισμένους περιορισμούς.

Ο ρυθμός μετάδοσης δεδομένων και η διεκπεραιωτική ικανότητα (Bandwidth and Throughput)

Το LoRaWAN υποστηρίζει χαμηλούς ρυθμούς δεδομένων, που κυμαίνονται συνήθως από 0,3 kbps έως και 50 kbps. Αυτό το καθιστά ακατάλληλο για εφαρμογές που απαιτούν υψηλό εύρος ζώνης, όπως η μετάδοση βίντεο ή η αποστολή μεγάλων αρχείων. Επιπλέον, οι ρυθμιστικοί φορείς συχνά επιβάλλουν περιορισμούς στον κύκλο εργασίας στις ζώνες ISM που χρησιμοποιεί το LoRaWAN, περιορίζοντας το χρόνο που μπορεί να εκπέμψει μια συσκευή. Αυτό μειώνει την αποτελεσματική διεκπεραιωτική ικανότητα και συχνότητα μεταδόσεων.

Καθυστέρηση (Latency)

Λόγω του χαμηλού ρυθμού δεδομένων και των περιορισμών του κύκλου εργασίας, το LoRaWAN εμφανίζει υψηλότερη καθυστέρηση σε σύγκριση με άλλες ασύρματες τεχνολογίες επικοινωνίας, όπως το Wi-Fi ή τα τηλεφωνικά δίκτυα. Αυτό το καθιστά ακατάλληλο για εφαρμογές σε πραγματικό χρόνο που απαιτούν άμεση ανταπόκριση. Επιπλέον, το LoRaWAN χρησιμοποιεί ασύγχρονο μοντέλο επικοινωνίας, που σημαίνει ότι οι συσκευές μεταδίδουν δεδομένα μόνο όταν το χρειάζονται, γεγονός που συμβάλλει περαιτέρω στην καθυστέρηση.

Χωρητικότητα Δικτύου και Επεκτασιμότητα

Καθώς αυξάνεται ο αριθμός των συσκευών σε ένα δίκτυο LoRaWAN, αυξάνεται η πιθανότητα συγκρούσεων πακέτων και παρεμβολών, γεγονός που μπορεί να οδηγήσει σε απώλεια δεδομένων και μειωμένη απόδοση του δικτύου. Η χωρητικότητα του δικτύου περιορίζεται από το διαθέσιμο εύρος ζώνης και τους κανονισμούς του κύκλου εργασίας. Σε περιοχές με πυκνή συγκέντρωση συσκευών LoRaWAN, αυτό μπορεί να οδηγήσει σε συμφόρηση του δικτύου.

Ασφάλεια

Παρόλο που το LoRaWAN χρησιμοποιεί κρυπτογράφηση (AES-128), υπάρχουν ανησυχίες σχετικά με πιθανές ευπάθειες στη διαχείριση των κλειδιών και την ανθεκτικότητα του συστήματος κρυπτογράφησης. Η σωστή εφαρμογή και διαχείριση είναι κρίσιμες για τη διασφάλιση της ασφάλειας. Επιπλέον, οι συσκευές που αναπτύσσονται σε απομακρυσμένες ή δυσπρόσιτες τοποθεσίες μπορεί να είναι ευάλωτες σε φυσική παραβίαση, γεγονός που θα μπορούσε να θέσει σε κίνδυνο το δίκτυο.

Κατανάλωση Ενέργειας

Ενώ το LoRaWAN είναι σχεδιασμένο για χαμηλή κατανάλωση ενέργειας, οι συσκευές πρέπει να διαχειρίζονται προσεκτικά την ενέργεια, ειδικά κατά τη μετάδοση δεδομένων. Οι συχνές μεταδόσεις ή οι μεγαλύτερες αποστάσεις επικοινωνίας μπορούν να αδειάσουν την μπαταρία πιο γρήγορα. Επιπλέον, οι συσκευές συνήθως είναι σε κατάσταση ύπνου για εξοικονόμηση ενέργειας, γεγονός που σημαίνει ότι πρέπει να ξυπνήσουν για να λάβουν ή να στείλουν δεδομένα, εισάγοντας καθυστέρηση και ενδεχομένως υψηλότερη κατανάλωση ενέργειας κατά τη διάρκεια αυτών των περιόδων.

Ποιότητα Υπηρεσίας (QoS)

Λόγω της φύσης του καναλιού επικοινωνίας και των περιορισμών του κύκλου εργασίας, η ποιότητα υπηρεσίας μπορεί να είναι απρόβλεπτη. Αυτό είναι ιδιαίτερα κρίσιμο για εφαρμογές που απαιτούν συνεπή παράδοση δεδομένων. Σε ορισμένες λειτουργίες, το LoRaWAN δεν παρέχει μηχανισμό επιβεβαίωσης, οδηγώντας σε πιθανή απώλεια δεδομένων χωρίς τη δυνατότητα ανίχνευσής της.

Ρυθμιστικοί Περιορισμοί

Το LoRaWAN λειτουργεί σε μη αδειοδοτημένες ζώνες ISM, οι οποίες υπόκεινται σε ρυθμιστικούς περιορισμούς που διαφέρουν ανά περιοχή. Αυτό μπορεί να επηρεάσει την ανάπτυξη και την απόδοση του δικτύου. Επιπλέον, οι ρυθμιστικοί περιορισμοί στην ισχύ μετάδοσης μπορούν να επηρεάσουν τη μέγιστη εμβέλεια και την ικανότητα διείσδυσης του δικτύου.

Διαχείριση Συσκευών και Δικτύου

Η διαχείριση μεγάλου αριθμού συσκευών, συμπεριλαμβανομένης της προμήθειας, της ενημέρωσης του firmware και της διασφάλισης της ασφάλειας, μπορεί να είναι πολύπλοκη και απαιτητική σε πόρους. Επίσης, απαιτείται αποτελεσματικός συντονισμός μεταξύ των πυλών και των διακομιστών δικτύου για τη διαχείριση της συνδεσιμότητας των συσκευών και της δρομολόγησης των δεδομένων με αποδοτικό τρόπο.

Περιορισμοί του κύκλου λειτουργίας

Οι περιορισμοί αυτοί τίθενται κυρίως από ρυθμιστικές αρχές για να εξασφαλίσουν την δίκαια χρήση του διαθέσιμου φάσματος στις συχνότητες ISM (industrial, scientific medical). Οι περιορισμοί ορίζουν το χρονικό διάστημα που μία συσκευή μπορεί να διαμείνει σε ένα κανάλι για να αποτρέψει τις παρεμβολές και τη συμφόρηση.

Ευρώπη (863-870 MHz)

Στην Ευρώπη, το LoRaWAN χρησιμοποιεί 24 με 80 κανάλια είτε με 125kHz είτε με 250kHz εύρος ζώνης, το εύρος ζώνης των 250kHz μπορεί επιτευχθεί χρησιμοποιώντας DR6 (Data Rate). Όλες οι τερματικές συσκευές LoRaWAN υποχρεούνται να λειτουργούν τουλάχιστον στα κανάλια των συχνοτήτων : 868.10 MHz, 868.30 MHz και 868.50 MHz. Αυτά τα κανάλια πρέπει να έχουν εύρος ζώνης 125 kHz σε DR0 ως και DR5, και να είναι περιορισμένα σε κύκλο λειτουργίας 1%. Αυτά τα 3 κανάλια χρησιμοποιούνται για την μετάδοση Join requests, οπότε όλες οι πύλες πρέπει επίσης να ακούν σε αυτά τα κανάλια.

Η LoRa Alliance συνιστά έναν περιορισμό κύκλου λειτουργίας (duty cycle) 1% στη ζώνη συχνοτήτων της Ευρώπης, που σημαίνει ότι μια συσκευή μπορεί να εκπέμπει για όχι παραπάνω από το 1% του χρόνου, διασφαλίζοντας παράλληλα τη μέγιστη EIRP (Effective Isotropic Radiated Power) των +16 dBm. Ο σκοπός του ορίου του κύκλου λειτουργίας είναι να εξασφαλίσει ότι οι συσκευές που λειτουργούν στις συχνότητες ISM δεν προκαλούν βλαβερές παρεμβολές σε άλλες συσκευές που δουλεύουν στις ίδιες συχνότητες. Το EIRP των 16 dBM είναι ίσο με 40 mW και είναι αντίστοιχο με ERP των 14 dBm, το οποίο είναι ίσο με 25 mW.

Το ETSI (European Telecommunications Standards Institute) έχει διαχωρίσει την Ευρωπαϊκή συχνότητα σε 6 ύπο συχνότητες (K, L, M, N, P, Q) και επιβάλει περαιτέρω περιορισμούς στον κύκλο λειτουργίας και το μέγιστο ERP για κάποιες από τις υποσυχνότητες.

- K (863 MHz - 865 MHz): 0.1%, 25 mW ERP
- L (865 MHz - 868 MHz): 1%, 25 mW ERP
- M (868 MHz - 868.6 MHz): 1%, 25 mW ERP

- N (868.7 MHz - 869.2 MHz): 0.1%, 25 mW ERP
- P (869.4 MHz - 869.65 MHz): 10%, 500 mW ERP
- Q (869.7 MHz - 870 MHz): 1%, 25 mW ERP

Για παράδειγμα, οι συσκευές που λειτουργούν στο εύρος 869.4 - 869.65 MHz, το οποίο είναι η υποσυχνότητα P, έχει περιορισμό κύκλου λειτουργίας 10% του χρόνου. Συνήθως, οι περιορισμοί των υπο συχνοτήτων K, L, M, N και Q εφαρμόζονται στις μεταδόσεις uplinks και οι περιορισμοί της υπο συχνότητας P αφορούν τις μεταδόσεις downlink. Συνήθως, ο αριθμός τερματικών συσκευών σε ένα LoRaWAN δίκτυο είναι πολύ μεγαλύτερος από τον αριθμό των πυλών, το οποίο σημαίνει ότι οι πύλες μπορούν να φτάσουν το όριο του κύκλους λειτουργίας τους πολύ πιο γρήγορα από τις τερματικές συσκευές. Το κανάλι RX2 του LoRaWAN συνήθως επιλέγεται να βρίσκεται στην υποσυχνότητα P προκειμένου να επωφεληθεί απο τον υψηλότερο περιορισμό του κύκλου λειτουργίας και να παρέχει 10 φορές περισσότερες λήψεις προς τις πύλες αντί για τις τερματικές συσκευές.

Εξ ορισμού, το παράθυρο λήψεων downlink του RX1 έχει την ίδια συχνότητα με το προηγούμενο uplink, ενώ το παράθυρο λήψεων downlink του RX2 έχει μια σταθερή συχνότητα, η προεπιλογή είναι 869.525 MHz σε DR0 χρησιμοποιώντας SF12 με 125 kHz. Στην Ευρώπη τα 869.525 MHz χρησιμοποιούνται για συσκευές κλάσης B και πιο συγκεκριμένα για μεταδόσεις beacon και ping slots.

Αμερική (902-928 MHz)

Στην Αμερική, Καναδά και όλες τις υπόλοιπες περιοχές που είναι μέρος του ITU region 2 η συχνότητα ISM είναι χωρισμένη σε 80 κανάλια, 72 για uplinks και 8 για downlinks. Υπάρχουν 64 uplink κανάλια, αριθμημένα απο 0 εως 63, το καθένα με εύρος ζώνης των 125 kHz. Αυτά τα κανάλια ξεκινάνε σε μία κεντρική συχνότητα 902.3 MHz και αυξάνονται γραμμικά ανά 200 kHz μέχρι και στα 914.9 MHz. Υπάρχουν 8 uplink κανάλια, αριθμημένα απο 64 μέχρι και 71, το καθένα με εύρος ζώνης των 500 kHz. Αυτά τα κανάλια ξεκινάνε σε μια κεντρική συχνότητα στα 903.0 MHz και αυξάνονται γραμμικά ανά 1.6 MHz μέχρι και τα 914.2 MHz. Υπάρχουν 8 downlink κανάλια, αριθμημένα απο 0 εως και 7, το καθένα με εύρος ζώνης των 500 kHz. Αυτά τα κανάλια ξεκινάνε σε μια κεντρική συχνότητα των 923.3 MHz και αυξάνονται γραμμικά ανα 600kHz μέχρι και τα 927.5 MHz.

Οι τερματικές συσκευές πρέπει να έχουν την δυνατότητα να αποθηκεύσουν παραμέτρους για τα 72 αυτά κανάλια. Ο χρόνος παραμονής (dwell time) για το LoRaWAN στις συχνότητες της Αμερικής αναφέρεται στο χρονικό διάστημα που μία συσκευή σπαταλάει σε ένα συγκεκριμένο κανάλι πριν αλλάξει στο επόμενο κανάλι. Ο χρόνος παραμονής είναι μια σημαντική παράμετρος για τις συσκευές LoRaWAN για να εξασφαλίσουν ότι οι συσκευές μπορούν να μεταδώσουν δεδομένα επιτυχημένα ενώ αποφεύγουν τις παρεμβολές με άλλες συσκευές.

Για την συχνότητα ISM της Αμερικής, ο χρόνος παραμονής για την μετάδοση ενός LoRaWAN μηνύματος δεν μπορεί να ξεπερνάει τα 400 ms. Είναι σημαντικό να λαμβάνεται υπόψην αυτός ο περιορισμός στον χρόνο παραμονής όταν κάποιος σχεδιάζει και χειρίζεται τέτοιες συσκευές. Για παράδειγμα, αν μια τέτοια συσκευή μεταδίδει μηνύματα σε μεγάλο ρυθμό δεδομένων, τότε ίσως να πρέπει να γίνει χρήση γρηγορότερης αλλαγής καναλιών για να τηρείται ο περιορισμός των 400 ms χρόνου παραμονής.

Οι τερματικές συσκευές όσο και οι πύλες είναι περιορισμένες σε ένα μέγιστο +30 dBm. Το μέγιστο EIRP μπορεί διατηρηθεί προσαρμόζοντας την ισχύ μετάδοσης, την απολαβή κεραίας και τη χασούρα καλωδίων. Το κανάλι για το παράθυρο λήψης του RX1 βασίζεται στο κανάλι uplink που χρησιμοποιήθηκε για να σταλεί το προηγούμενο uplink. Η προεπιλογή για το κανάλι downlink για το παράθυρο λήψης RX2 είναι στα 923.3 MHz με ρυθμό δεδομένων DR8. Οι συσκευές κλάσης B χρησιμοποιούν τις συχνότητες 923.3, 923.9, 924.5, 925.1, 925.7, 926.3, 926.9, και 927.5 MHz για τις μεταδόσεις beacon και τα ping slots. Αυτές οι συχνότητες αλλάζουν περιστροφικά και ξεκινόντας στα 923.3 MHz.

2.9. The Things Network

Στην δική μας περίπτωση θα χρησιμοποιήσουμε τα συστήματα και τις υπηρεσίες της The Things Network (TTN). Το The Things Network είναι μια ανοιχτή, συλλογική και αποκεντρωμένη υποδομή δικτύωσης για εφαρμογές IoT. Επιτρέπει σε συσκευές να επικοινωνούν με διαδίκτυο χρησιμοποιώντας χαμηλής ισχύος, μακράς εμβέλειας σήματα ραδιοσυχνοτήτων. Το TNN παρέχει μια εναλλακτική λύση σε παραδοσιακές μεθόδους σύνδεσης όπως το WiFi ή τα κινητά δίκτυα, ιδιαίτερα για συσκευές IoT που χρειάζεται να λειτουργούν σε απομακρυσμένες περιοχές ή όπου η κατανάλωση ενέργειας είναι λόγος ανησυχίας.

Το δίκτυο βασιζεται στο πρωτόκολλο LoRaWAN, το οποίο, όπως είδαμε παραπάνω επιτρέπει την επικοινωνία μεγάλης απόστασης μεταξύ συσκευών και πυλών. Το TNN λειτουργεί με την αρχή υποδομής που οδηγείται από την κοινότητα, όπου άτομα, οργανισμοί και επιχειρήσεις εγκαθιστούν πύλες για να επεκτείνουν την κάλυψη του δικτύου. Αυτές οι πύλες επικοινωνούν με τι συσκευές, συλλέγουν δεδομένα και τα προωθούν στο νέφος όπου μπορούν να επεξεργαστούν και να αναλυθούν. Το TTN προσφέρει μια σειρά υπηρεσιών και εργαλείων για την υποστήριξη προγραμματιστών και χρηστών στην δημιουργία και την ανάπτυξη εφαρμογών IoT, συμπεριλαμβανομένων διεπαφών προγραμματιστή (APIs), τεκμηρίωσης και ενσωματώσεων με διάφορες πλατφόρμες IoT. Στοχεύει στη μείωση των φραγμών για την ανάπτυξη IoT εφαρμογών παρέχοντας μια ανοιχτή και προσβάσιμη υποδομή δικτύου.

Πιο πάνω μιλήσαμε για τους περιορισμούς που τίθενται στο πρωτόκολλο LoRaWAN, το TTN λόγω του ότι είναι δημόσιο δίκτυο, θέτει κι αυτό μερικούς περιορισμούς για την δίκαια χρήση του δικτύου και την αποφυγή παρεμβολών και συγκρούσεων των πακέτων δεδομένων που μεταδίδονται στο δίκτυο. Οι περιορισμοί αυτοί ονομάζονται Fair Access Policy, θέτει όριο στα μηνύματα uplink τα οποία δεν μπορούν να έχουν πάνω από 30 δευτερόλεπτα time on air ανα 24 ώρες και ισχύει για κάθε end device και το όριο των downlink μηνυμάτων είναι στα 10 μηνύματα ανα 24 ώρες και ισχύει επίσης για κάθε end device. Αυτοί οι περιορισμοί δεν υπάρχουν αν γίνει χρήση ιδιωτικού δικτύου LoRaWAN.

Ιδιωτικά LoRaWAN Network Stacks

Το ChirpStack προσφέρει μια αξιόπιστη, επεκτάσιμη και προσαρμόσιμη λύση για τη λειτουργία δικτύων LoRaWAN, ως μια ανοιχτού κώδικα network stack LoRaWAN. Από την πρώτη του κυκλοφορία ως LoRaServer, το ChirpStack έχει αυξήσει τη δημοτικότητά του χάρη στο εκτεταμένο σύνολο δυνατοτήτων του και την προσέγγιση ανάπτυξης που καθοδηγείται

από την κοινότητα. Ο ChirpStack Network Server, ο ChirpStack Application Server και το ChirpStack Gateway Bridge είναι μεταξύ των πολλών μερών του, το καθένα έχει μια διακριτή λειτουργία στην αρχιτεκτονική του δικτύου. Το ChirpStack είναι κατάλληλο για ένα ευρύ φάσμα εφαρμογών, από μικρά ιδιωτικά δίκτυα μέχρι μεγάλες εμπορικές εγκαταστάσεις, χάρη στον αρθρωτό του σχεδιασμό, που επιτρέπει στους χρήστες να προσαρμόζουν τη διαμόρφωσή τους ανάλογα με τις δικές τους ανάγκες.

Η ευελιξία ανάπτυξης του ChirpStack είναι ένα από τα καλύτερα χαρακτηριστικά του. Το ChirpStack μπορεί να αναπτυχθεί στο cloud για καλύτερη επεκτασιμότητα και ευκολία διαχείρισης, ή να φιλοξενηθεί τοπικά, παρέχοντας στους χρήστες πλήρη έλεγχο των δεδομένων και των διαμορφώσεων του δικτύου τους. Η πλατφόρμα επιτρέπει την ομαλή επεξεργασία και ροή δεδομένων υποστηρίζοντας μια σειρά συνδέσεων με γνωστές βάσεις δεδομένων, υπηρεσίες cloud και εξωτερικές εφαρμογές. Το ChirpStack εγγυάται επίσης συμβατότητα με μια ευρεία γκάμα συσκευών και πύλων LoRaWAN, τηρώντας τις προδιαγραφές του LoRaWAN πρωτοκόλλου, που προάγει τη διαλειτουργικότητα και μειώνει τις ανησυχίες για εξάρτηση από συγκεκριμένους προμηθευτές.

Η κοινότητα του ChirpStack είναι ενεργή και υποστηρικτική, συμμετέχοντας ενεργά στις συνεχείς αναβαθμίσεις και βελτιώσεις της πλατφόρμας. Προσφέρεται μια ποικιλία από εκπαιδευτικά προγράμματα και άφθονη τεκμηρίωση για να βοηθήσουν τους χρήστες να ξεκινήσουν και να επιλύσουν τυπικά προβλήματα. Το ChirpStack μπορεί να παραμένει στην αιχμή των λύσεων διακομιστών δικτύου LoRaWAN χάρη σε αυτό το συνεργατικό περιβάλλον που ενθαρρύνει την καινοτομία. Η κατασκευή και η διαχείριση αποτελεσματικών δικτύων IoT γίνεται πιο εύκολη με το ChirpStack, μια αξιόπιστη και ευπροσάρμοστη πλατφόρμα που μπορεί να χρησιμοποιηθεί για βιομηχανικά, ακαδημαϊκά ή έξυπνα έργα πόλεων.

Οι κύριες διαφορές του TTN με το ChirpStack ή οποιοδήποτε άλλο ιδιωτικό LoRaWAN Network Stack είναι ότι το TTN προσφέρει ένα αρκετά μεγάλο δίκτυο στο οποίο μπορεί να έχει πρόσβαση ο καθένας, αν είναι αρκετά τυχερός να είναι στην εμβέλεια κάποιου gateway που έχει εγκατασταθεί από κάποιον άλλο χρήστη του TNN δικτύου αλλιώς θα πρέπει να διαθέσει δικό του gateway. Όντας ένα δημόσιο δίκτυο όμως το TTN έχει όπως έχουμε πεί κάποιους περιορισμούς που το καθιστούν μη πρακτικό για project μεσαίου προς μεγάλου μεγέθους. Τα ιδιωτικά δίκτυα όπως το ChirpStack προσφέρουν μεγαλύτερη άνεση από άποψη περιορισμών και επεκτασιμότητας διότι η εγκατάσταση γίνεται από τον ίδιο τον χρήστη του δικτύου αλλά με το αρνητικό ότι το κόστος είναι μεγαλύτερο διότι θα χρειαστεί να γίνει η εγκατάσταση πολλαπλών gateways έτσι ώστε να γίνει κάλυψη της περιοχής έτσι ώστε να μπορούν τα end devices να επικοινωνούν με το δικτυο.

Μια ακόμα μεγάλη διαφορά είναι ότι το TTN πρέπει να χρησιμοποιηθεί στο cloud ή με ένα μηνιαίο κόστος μπορεί να γίνει η χρήση του σε μηχανήματα server του χρήστη. Το ChirpStack προσφέρει την δυνατότητα cloud αλλά το μεγαλύτερο του πλεονέκτημα είναι ότι λόγω της open source φύσης του το network stack είναι δωρεάν προς όλους και ο καθένας μπορεί να το χρησιμοποιήσει σε δικο του server.

2.10. MQTT

Το MQTT (αρχικά ακρωνύμιο του Message Queuing Telemetry Transport) είναι ένα ελαφρύ, publish/subscribe, πρωτόκολλο δικτύου μηχανής προς μηχανή για υπηρεσία ουράς μηνυμάτων. Σχεδιάστηκε για συνδέσεις με απομακρυσμένες τοποθεσίες που έχουν συσκευές

με περιορισμένους πόρους ή περιορισμένο εύρος ζώνης δικτύου, όπως στο Internet of Things (IoT). Πρέπει να εκτελείται πάνω από ένα πρωτόκολλο μεταφοράς που παρέχει καθορισμένες, χωρίς απώλειες, αμφίδρομες συνδέσει, συνήθως TCP/IP, αλλά και πιθανώς πάνω από το QUIC. Είναι ένα ανοιχτό πρότυπο OASIS και μια σύσταση του ISO (ISO/IEC 20922). Παρόλο που η τεχνική επιτροπή ονομάζεται "OASIS Message Queuing Telemetry Transport Technical Committee," σε μεταγενέστερες εκδόσεις του πρωτοκόλλου απλώς αναφέρεται ως "MQTT". Από το 2013 και μετά το ακρωνύμιο MQTT σταμάτησε να σημαίνει κάτι.

Με ένα καταστατικό εγγυώντας ότι θα επιτραπούν μόνο μικρές αλλαγές στον ορισμό, η IBM υπέβαλε το MQTT v3.1 στο σώμα προδιαγραφών της OASIS το 2013. Στις 29 Οκτωβρίου 2014, η OASIS δημοσίευσε την έκδοση 3.1.1 του προτύπου μετά την ανάληψη της συντήρησής του από την IBM. Στις 7 Μαρτίου 2019, δημοσιεύτηκε μια πιο σημαντική ενημέρωση του MQTT έκδοσης 5 με αρκετά νέα χαρακτηριστικά. Μια παραλλαγή του κύριου πρωτοκόλλου που ονομάζεται MQTT-SN (MQTT για Δίκτυα Αισθητήρων) σχεδιάστηκε για ενσωματωμένες συσκευές με μπαταρία σε μη δίκτυα TCP/IP όπως το Zigbee.

Το πρωτόκολλο MQTT ορίζει δύο κατηγορίες οντοτήτων δικτύου: ένα μεσίτη μηνυμάτων και πολλούς πελάτες. Ένας server που δέχεται όλα τα μηνύματα από τους clients και τα προωθεί στους σωστούς προορισμούς clients ονομάζεται MQTT Broker. Κάθε συσκευή που χρησιμοποιεί βιβλιοθήκη MQTT και διαμορφώνει μια δικτυακή σύνδεση με ένα MQTT Broker θεωρείται MQTT Client, από έναν μικροελεγκτή έως έναν πλήρως λειτουργικό server. Τα θέματα οργανώνονται σε μια ιεραρχία πληροφοριών. Ένας publisher ειδοποιεί τον συνδεδεμένο broker χρησιμοποιώντας ένα μήνυμα ελέγχου που περιέχει τα νέα δεδομένα που θέλει να μοιραστεί. Στη συνέχεια, ο broker δίνει τις πληροφορίες σε οποιουσδήποτε clients έχουν κάνει subscribe σε αυτό το topic.

Δεν απαιτείται καμία πληροφορία για την ποσότητα ή τις τοποθεσίες των subscribers από τον publisher, και οι subscribers δεν απαιτούνται να διαμορφώσουν καμία πληροφορία σχετικά με τους publishers. Εκτός εάν ο publisher ορίσει το μήνυμα ως διατηρούμενο μήνυμα, ένας broker θα διαγράψει οποιοδήποτε μήνυμα λάβει για ένα topic για το οποίο δεν υπάρχουν ενεργοί subscribers. Ένα κανονικό μήνυμα MQTT με τη σημαία διατήρησης ρυθμισμένη σε true ονομάζεται διατηρούμενο μήνυμα. Για το επιλεγμένο topic, ο broker κρατάει το ιστορικό του πιο πρόσφατου μηνύματος που διατηρήθηκε καθώς και τη σχετική ποιότητα εξυπηρέτησης (QoS). Το διατηρούμενο μήνυμα στέλνεται σε κάθε client αμέσως μετά το subscription σε ένα πρότυπο topic που ταιριάζει με το θέμα του διατηρούμενου μηνύματος.

Για κάθε θέμα, ο broker αποθηκεύει ένα μόνο διατηρούμενο μήνυμα. Ως αποτέλεσμα, οι νέοι subscribers μπορούν να λάβουν τις πιο πρόσφατες πληροφορίες για ένα θέμα χωρίς να χρειάζεται να περιμένουν για την επόμενη ενημέρωση από τον publisher. Όταν ένας client publisher συνδέεται αρχικά στον broker, έχει τη δυνατότητα να διαμορφώσει ένα προεπιλεγμένο μήνυμα που θα παραδοθεί στους subscribers στην περίπτωση που ο broker παρατηρήσει μια απροσδόκητη αποσύνδεση από τον client publisher.

Οι clients επικοινωνούν αποκλειστικά με έναν broker. Ωστόσο, ένα σύστημα μπορεί να έχει πολλούς server brokers που ανταλλάσσουν πληροφορίες σύμφωνα με τα topics των ενεργών subscribers τους. Ακόμα και δύο bytes δεδομένων μπορούν να αποτελέσουν ένα βασικό μηνύματος ελέγχου MQTT. Εάν είναι απαραίτητο, ένα μήνυμα ελέγχου μπορεί να περιλαμβάνει σχεδόν 256 gigabytes δεδομένων.

Για να δημοσιευθούν δεδομένα, να επιβεβαιωθεί η παραλαβή δεδομένων, να συνδεθεί και να αποσυνδεθεί ένας client από ένα broker και να διαχειριστεί η σχέση μεταξύ του client και του server, υπάρχουν δεκατέσσερα διακριτά είδη μηνυμάτων. Η μετάδοση δεδομένων με MQTT βασίζεται στο πρωτόκολλο TCP. Το MQTT-SN, μια παραλλαγή, προτιμάται έναντι άλλων μεταφορών όπως το Bluetooth ή το UDP. Το MQTT μεταδίδει διαπιστευτήρια σύνδεσης σε μη κρυπτογραφημένο κείμενο χωρίς κανένα μηχανισμό ασφάλειας ή πιστοποίησης. Για να γίνει αυτό, τα δεδομένα που επικοινωνούνται μπορούν να κρυπτογραφηθούν και να προστατευτούν από παραποίηση, αλλοίωση και ανακάλυψη χρησιμοποιώντας TLS. Το 1883 είναι η προεπιλεγμένη θύρα MQTT χωρίς κρυπτογράφηση. Το 8883 είναι η κρυπτογραφημένη θύρα.

MQTT Broker

Ένα πρόγραμμα υπολογιστή που τρέχει σε τοπικο μηχάνημα server ή στο cloud είναι γνωστό ως MQTT Broker. Μπορεί να αναπτυχθεί εσωτερικά ή να φιλοξενείται από έναν τρίτο φορέα. Υπάρχουν ιδιόκτητες και ανοικτού κώδικα εκδόσεις του. Ο broker λειτουργεί ως υπηρεσία ταχυδρομείου. Οι clients MQTT χρησιμοποιούν topics αντί της άμεσης σύνδεσης στην διεύθυνση του αναμενόμενου αποδέκτη. Κάθε subscriber λαμβάνει ένα αντίγραφο κάθε επικοινωνίας που σχετίζεται με αυτό το topic. Ένας μόνο client μπορεί να κάνει subscriptions σε topics με πολλούς brokers , και πολλοί clients μπορούν να εγγραφούν σε ένα topic από έναν broker.

Επειδή το MQTT είναι ένα διπλής κατεύθυνσης πρωτόκολλο επικοινωνίας, κάθε client μπορεί να δημιουργήσει και να λάβει δεδομένα τόσο με το publishing όσο και με το subscription. Με άλλα λόγια, οι συσκευές μπορούν να μεταδίδουν δεδομένα αισθητήρων και να λαμβάνουν ταυτόχρονα δεδομένα διαμόρφωσης ή εντολές ελέγχου. Αυτό διευκολύνει τη διαχείριση και τον έλεγχο των συσκευών καθώς και την ανταλλαγή δεδομένων. Ένας client πρέπει να δημοσιεύει πολλά μηνύματα στον broker, καθένα με ένα διακριτό topic που καθορίζεται. Δεν είναι δυνατό για έναν client να μεταδώσει τα ίδια δεδομένα σε πολλά topics ταυτόχρονα.

Οι συσκευές clients και η εφαρμογή server διαχωρίζονται χρησιμοποιώντας την αρχιτεκτονική MQTT broker. Με αυτόν τον τρόπο, οι clients διατηρούνται αδιάφοροι για τις πληροφορίες των άλλων. Εάν διαμορφωθεί σωστά, το MQTT μπορεί να χρησιμοποιήσει κρυπτογράφηση TLS σε συνδέσεις που είναι ασφαλισμένες με κωδικό πρόσβασης, όνομα χρήστη και πιστοποιητικό. Οι clients και το λογισμικό broker έχουν τη δυνατότητα να μεταβούν αυτόματα σε έναν εφεδρικό ή αυτόματο αντίγραφο ασφαλείας broker σε περίπτωση διακοπής. Τα αντίγραφα ασφαλείας μπορούν επίσης να διαμορφωθούν για να διανείμουν τα φορτία των clients μεταξύ πολλών server επί τόπου, cloud servers ή μιας συνδυασμένης λύσης από αυτά.

Ο Broker υποστηρίζει στανταρ MQTT και MQTT που ακολουθούν προδιαγραφές, όπως το Sparkplug . Ο ίδιος server, την ίδια στιγμή και με τα ίδια πρότυπα ασφάλειας, μπορεί να χρησιμοποιηθεί για αυτό. Μια λειτουργία που είναι γνωστή ως "διαρκείς συνεδρίες"(persistent sessions) επιτρέπει στο broker να καταγράφει όλα τα δεδομένα συνεδρίας ακόμα και όταν η συσκευή είναι απενεργοποιημένη. Σε αυτήν την κατάσταση, ένας broker θα αποθηκεύει τόσο τις πληροφορίες σύνδεσης για κάθε client, τα topics στα οποία έχει εγγραφεί ο κάθε client, όσο και οποιαδήποτε μηνύματα για ένα topic με QoS 1 ή 2.

Τα κύρια πλεονεκτήματα του MQTT Broker είναι :

- Ξεφορτώνεται από τις ευάλωτες και ανασφαλείς συνδέσεις, αν ρυθμιστεί για αυτό.
- Μπορεί να επεκταθεί πολύ εύκολα από μια συσκευή σε χιλιάδες.
- Διαχειρίζεται και παρακολουθεί την κατάσταση σύνδεσης όλων των clients, συμπεριλαμβάνοντας τα διαπιστευτήρια ασφαλείας και τα πιστοποιητικά, αν ρυθμιστεί για αυτό.
- Μειώνει το φορτίο το δικτύου χωρίς να θέτει σε κίνδυνο την ασφάλεια, αν ρυθμιστεί για αυτό.

Τύποι μηνυμάτων

- Connect - Περιμένει να εδραιωθεί μια σύνδεση με τον server και δημιουργεί έναν σύνδεσμο μεταξύ των κόμβων.
- Disconnect - Περιμένει τον MQTT client να ολοκληρώσει οποιαδήποτε εργασία πρέπει να εκτελέσει και την διακοπή της σύνδεσης TCP/IP.
- Publish - Επιστρέφει αμέσως στο thread της εφαρμογης μετά την παράδοση τους αιτήματος στον MQTT client.

QoS

Κάθε σύνδεση στον Broker μπορεί να καθορίσει ένα μέτρο QoS. Αυτά κατατάσσονται σε αύξουσα σειρά επιβάρυνσης:

1. Το πολύ μια φορά - το μήνυμα αποστέλλεται μόνο μια φορά και ο client και ο broker δεν αναλαμβάνουν επιπλέον μέτρα για την επιβεβαίωση της παράδοσης (fire and forget)
2. Τουλάχιστον μια φορά - το μήνυμα επαναλαμβάνεται απο τον τον αποστολέα πολλές φορές μέχρι να λάβει επιβεβαίωση (acknowledged delivery)
3. Ακριβώς μια φορά - ο αποστολέας και ο παραλήπτης διεξάγουν μια διαδικασία διασφάλισης για να εξασφαλιστεί ότι λαμβάνεται μόνο ένα αντίγραφο του μηνύματος (assured delivery)

Αυτό το πεδίο δεν επηρεάζει την επεξεργασία των υποβόσκων μεταδόσεων TCP, χρησιμοποιείται μόνο μεταξύ των αποστολέων και των παραληπτών MQTT.

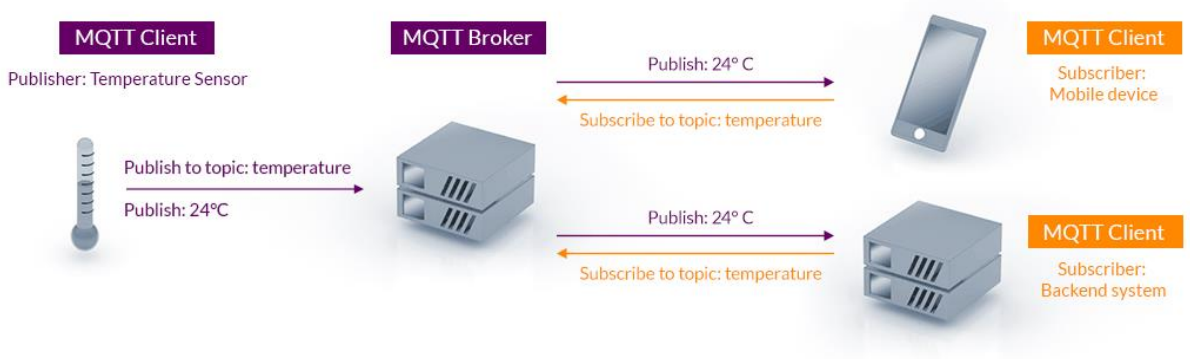


Fig. 7, MQTT Publish / Subscribe Architecture (source : https://mqtt.org/)

2.11. NanoMQ

Το NanoMQ είναι ένας εξαιρετικά ελαφρύς MQTT Broker (Message Queuing Telemetry Transport) σχεδιασμένος για να διαχειρίζεται μηνύματα υψηλής απόδοσης και χαμηλής καθυστέρησης. Είναι χτισμένος πάνω στην πλατφόρμα NNG (Nanomsg Next Generation), μια βιβλιοθήκη υψηλής απόδοσης για ανταλλαγή μηνυμάτων, που επιτρέπει στο NanoMQ να υποστηρίζει διάφορα πρωτόκολλα επικοινωνίας με αποδοτικότητα. Ο κύριος στόχος του NanoMQ είναι να καλύψει τις ανάγκες των εφαρμογών IoT (Internet of Things) παρέχοντας έναν αξιόπιστο, επεκτάσιμο και αποδοτικό broker που μπορεί να λειτουργεί σε συσκευές με περιορισμένους πόρους. Το μικρό του αποτύπωμα και η υψηλή του απόδοση το καθιστούν ιδανική επιλογή για περιβάλλοντα edge computing όπου οι πόροι είναι περιορισμένοι, αλλά η αξιόπιστη και γρήγορη επικοινωνία είναι κρίσιμη.

Ένα από τα κύρια χαρακτηριστικά του NanoMQ είναι η υποστήριξη του MQTT 5.0, της τελευταίας έκδοσης του πρωτοκόλλου MQTT, που εισάγει αρκετές βελτιώσεις σε σύγκριση με προηγούμενες εκδόσεις. Αυτές οι βελτιώσεις περιλαμβάνουν καλύτερη αναφορά σφαλμάτων, καλύτερη υποστήριξη για μεγάλης κλίμακας εφαρμογές και βελτιωμένα χαρακτηριστικά ασφάλειας. Επιπλέον, το NanoMQ υποστηρίζει μια σειρά από προηγμένες λειτουργίες όπως το MQTT over WebSocket, τη γεφύρωση με άλλους MQTT Brokers και την ενσωμάτωση με διάφορα συστήματα αποθήκευσης και επεξεργασίας δεδομένων. Αυτές οι δυνατότητες το καθιστούν εξαιρετικά ευέλικτο και κατάλληλο για ποικίλα σενάρια ανάπτυξης, από απλά δίκτυα αισθητήρων μέχρι σύνθετα συστήματα αυτοματισμού βιομηχανίας.

Το NanoMQ δίνει επίσης έμφαση στην ευκολία χρήσης και την ευελιξία. Παρέχει ένα πλούσιο σύνολο επιλογών διαμόρφωσης, επιτρέποντας στους χρήστες να προσαρμόσουν τον broker στις συγκεκριμένες τους ανάγκες. Ο broker είναι σχεδιασμένος να είναι επεκτάσιμος, με μια αρχιτεκτονική plugins που επιτρέπει στους προγραμματιστές να προσθέτουν προσαρμοσμένες λειτουργίες όταν χρειάζεται. Επιπλέον, η απόδοση του NanoMQ έχει δοκιμαστεί διεξοδικά, αποδεικνύοντας την ικανότητά του να διαχειρίζεται δεκάδες χιλιάδες ταυτόχρονες συνδέσεις με χαμηλή καθυστέρηση. Αυτό το καθιστά αξιόπιστη επιλογή για εφαρμογές κρίσιμης σημασίας όπου η έγκαιρη και αποδοτική παράδοση δεδομένων είναι καίρια. Συνολικά, το NanoMQ αντιπροσωπεύει ένα ισχυρό εργαλείο στο οικοσύστημα του IoT, συνδυάζοντας τα οφέλη του lightweight σχεδιασμού με την ισχυρή απόδοση και την εκτεταμένη υποστήριξη χαρακτηριστικών.

3. Σχεδιασμός

3.1. Ανάλυση απαιτήσεων

Σειριοποίηση δεδομένων για την ώρα της μετάδοσης

Τα δεδομένα που συλλέγονται από τους αισθητήρες RS-485 πρέπει να σειριοποιούνται χρησιμοποιώντας protocol buffers πρίν την μετάδοση τους μέσω του LoRaWAN δικτύου. Οι σειριοποίηση πρέπει να περιλαμβάνει όλες τις απαραίτητες πληροφορίες, όπως το ID του αισθητήρα, την ώρα μέτρησης σε μορφή unix timestamp, και τις μετρήσεις του αισθητήρα, όπως η υγρασία εδάφους, θερμοκρασία και την αγωγιμότητα του εδάφους. Ο backend server που είναι γραμμένος σε nodejs θα μπορεί να κάνει αποσειριοποίηση των δεδομένων χρησιμοποιώντας το .proto αρχείο και την βιβλιοθήκη protobufjs για την αποθήκευση και την επεξεργασία των δεδομένων.

Το σύστημα θα πρέπει να είναι σε θέση να χειρίζεται διάφορες εκδόσεις των μηνυμάτων protocol buffers, έτσι εξασφαλίζοντας backwards compatibility με παλαιότερες εκδόσεις των μηνυμάτων. Επίσης θα πρέπει να επιτρέπουν στην εύκολη προσθήκη νέων τύπων δεδομένων και πεδίων στο μέλλον, χωρίς να διακόπτεται η λειτουργία του συστήματος. Η σειριοποίηση και αποσειριοποίηση των δεδομένων θα πρέπει να είναι αποδοτική με ελάχιστο χρόνο επεξεργασίας, τα μηνύματα που δημιουργούνται με τα protocol buffers πρέπει να είναι όσο το δυνατόν πιο συμπιεσμένα για την μείωση του όγκου των δεδομένων που μεταδίδονται με LoRaWAN.

Ασύρματη επικοινωνία και LoRaWAN

Τα δεδομένα που συλλέγονται από τους αισθητήρες RS-485 πρέπει να αποστέλονται ασύρματα μέσω του δικτύου LoRaWAN στον server. Το σύστημα πρέπει να υποστηρίζει αμφίδρομη επικοινωνία, επιτρέποντας την αποστολή εντολών από τον server στους ενεργοποιητές. Το σύστημα πρέπει να είναι συμβατό με το πρότυπο LoRaWAN και να υποστηρίζει τις προδιαγραφές που ορίζονται για τις διάφορες κατηγορίες συσκευών (class A, B, C). Πρέπει να χρησιμοποιούνται οι κατάλληλες συχνότητες και κανάλια σύμφωνα με τις τοπικές κανονιστικές απαιτήσεις.

Αισθητήρες και ενεργοποιητές RS-485

Οι αισθητήρες και οι ενεργοποιητές πρέπει να είναι συμβατοί με το πρωτόκολλο επικοινωνίας RS-485 για μετάδοση δεδομένων. Το σύστημα πρέπει να υποστηρίζει τη σύνδεση και επικοινωνία με πολλαπλούς αισθητήρες RS-485. Οι αισθητήρες πρέπει να μπορούν να ανιχνεύσουν αλλαγές στο έδαφος ή σε υπόστρωμα, αλλαγές θερμοκρασία, υγρασίας κλπ. Τα δεδομένα από τους αισθητήρες πρέπει να συλλέγονται και να αποστέλλονται στο κεντρικό σύστημα μέσω του LoRaWAN δικτύου. Το σύστημα πρέπει να μπορεί να ανιχνεύει και να ανταποκρίνεται σε δεδομένα από τους αισθητήρες, όπως αλλαγές στη θερμοκρασία, την

υγρασία κλπ. Πρέπει να υποστηρίζει την αποστολή εντολών στους ενεργοποιητές με βάση τα δεδομένα που συλλέγονται από τους αισθητήρες. Το σύστημα πρέπει να υποστηρίζει την αξιόπιστη μετάδοση δεδομένων έως και τα 1200 μέτρα, όπως προβλέπει το πρότυπο RS-485.

3.2. Σχεδιασμός του συστήματος

Η αρχιτεκτονική του συστήματος για το σύστημα παρακολούθησης και ελέγχου φυτών είναι σχεδιασμένη να ενσωματώνει πολλά εξαρτήματα απρόσκοπτα για την εξασφάλιση της αποδοτικής συλλογής δεδομένων, επεξεργασίας και μετάδοσης. Τα κύρια κομμάτια του συστήματος είναι ένας μικροελεγκτής ο οποίος επικοινωνεί με τους αισθητήρες και τους ενεργοποιητές μέσω του RS-485 bus. Αυτοί οι μικροελεγκτές επικοινωνούν με τους RS-485 αισθητήρες μέσω τους σειριακού bus χρησιμοποιώντας ένα RS-485 to TTL converter. Οι μικροελεγκτές επικοινωνούν ασύρματα με την χρήση MQTT μηνυμάτων με το άλλο κύριο μέρος του συστήματος που είναι το LoRaWAN end device και ο MQTT Broker. Το end device χρησιμοποιεί LoRaWAN δίκτυο έτσι δίνοντας του την δυνατότητα για απομακρυσμένη επικοινωνία μεγάλης απόστασης.

Σχεδιασμός Protocol Buffers

Γίνεται χρήση των protocol buffers για την σειριοποίηση των δεδομένων που μεταδίδονται μέσω του LoRaWAN δικτύου, εξασφαλίζοντας αποδοτική και συμπαγής αναπαράσταση δεδομένων. Ο σχεδιασμός των protobuf περιλαμβάνει τον ορισμό ενός schema που προσδιορίζει την δομή των δεδομένων, συμπεριλαμβανομένου τον τύπο και τον αριθμό κάθε μεταβλητής. Το schema γίνεται compiled για να παραχθεί κώδικας ο οποίος μπορεί να χρησιμοποιηθεί για την σειριοποίηση και αποσειριοποίηση των μηνυμάτων. Η διαδικασία της σειριοποίησης είναι σημαντική για να μειωθεί το μέγεθος των δεδομένων, ο χρόνος της μετάδοσης και για την εξασφάλιση της ακεραιότητας των δεδομένων κατα την επικοινωνία.

Τα protobuf προσφέρουν μεγάλη ευκολία στη σειριοποίηση των δεδομένων σε αντίθεση με τους άλλους μεθόδους σειριοποίησης δεδομένων διότι το μόνο που χρειάζεται να γίνει είναι ο σχεδιασμός μιας δομής δεδομένων που χρειάζεται κάποιος και μετά από το γέμισμα της δομής αυτής, η μετατροπή της δομής σε hex values γίνεται αυτόματα με τη χρήση του αρχείου που δημιουργείται κατά το compilation του .proto αρχείου.

Για παράδειγμα αν κάποιος θα ήθελε να κάνει δική τους σειριοποίηση για τα δεδομένα θα έπρεπε να σχεδιάσει μια δομή η οποία θα πρέπει να είναι πρακτικά πάντα σταθερή και σε περίπτωση αλλαγής της δεν υπάρχει backwards ή forwards compatibility το οποίο σημαίνει ότι θα πρέπει να αλλάξει η δομή σε κάθε συσκευή που χρησιμοποιεί τη δομή αυτή. Ένα ακόμα αρνητικό είναι ότι η μετατροπή των δεδομένων σε hex values πρέπει να γίνει χειροκίνητα, που σημαίνει ότι θα πρέπει να γίνουν χειροκίνητα bit shifts και άλλοι υπολογισμοί για κάθε μεταβλητή της δομής ξεχωριστά για την σειριοποίηση και την αποσειριοποίηση.

Αισθητήρες και ενεργοποιητές RS-485

Η ενσωμάτωση των RS-485 αισθητήρων και ενεργοποιητών στο σύστημα παρέχει αξιοπιστία κατα τη συλλογή δεδομένων και δυνατότητα ελέγχου. Το RS-485 επιλέχθηκε λόγω του διαφορικού σήματος, το οποίο προσφέρει ανοσία στις παρεμβολές και επικοινωνία σε μεγάλες αποστάσεις. Οι αισθητήρες μετρούν διάφορες περιβαλλοντικές παραμέτρους όπως τη θερμοκρασία του χώματος, την υγρασία και άλλα, ενώ οι ενεργοποιητές ελέγχουν την άρδευση και άλλες αυτοματοποιημένες διαδικασίες. Η εγκατάσταση περιλαμβάνει την διαμόρφωση του διαύλου RS-485, την ανάθεση ξεχωριστών διευθύνσεων για για κάθε συσκευή, και την εξασφάλιση σωστών τερματικών συνδέσεων για την αποφυγή ανάκλασης σήματος και την αλλοίωση δεδομένων.

Επικοινωνία LoRaWAN

Η χρήση της τεχνολογίας LoRaWAN γίνεται διότι προσφέρει ασύρματη επικοινωνία σε μεγάλες αποστάσεις και με μικρή κατανάλωση ενέργειας, τα οποία την καθιστούν ιδανική για εφαρμογές απομακρυσμένης παρακολούθησης φυτών. Η αρχιτεκτονική του δικτύου LoRaWAN συμπεριλαμβάνει end devices, gateways που προωθεί τα δεδομένα στον network server, και έναν backend server ο οποίος επεξεργάζεται τα δεδομένα. Η ροή των δεδομένων είναι σχεδιασμένη για να εξασφαλίσει την αξιόπιστη μετάδοση από το μέρος της εγκατάστασης στο κεντρικό σύστημα, ακόμα και σε δύσκολες περιβαλλοντικές καταστάσεις. Το adaptive data rate και οι δυνατοί μηχανισμοί διόρθωσης σφαλμάτων βελτιώνει την αξιοπιστία και την αποδοτικότητα της επικοινωνίας.

Το end device χρειάζεται ένα μεσοδιάστημα τουλάχιστον 1 με 2 λεπτά για κάθε uplink που θέλει να κάνει, αυτό το μεσοδιάστημα αλλάζει και ανάλογα με το πόσο μεγάλο θα είναι το μήνυμα που θέλουμε να στείλουμε αλλα και ανάλογα με το spread factor που χρησιμοποιούμε. Αυτό είναι κυρίως πρόβλημα διότι χρησιμοποιούμε το δημόσιο δίκτυο της The Things Network, με την χρήση ιδιωτικού δικτύου αυτό το πρόβλημα θα εξαλειφθεί. Έχοντας αυτό κατα νού θα πρέπει να βάλουμε έναν χρόνο μετάδοσης στο end device που είναι πάνω από 2 λεπτά και το wifi node που φέρει τον αισθητήρα θα πρέπει επίσης να καταγράφει δεδομένα με τον αισθητήρα και να τα στέλνει στο end device κάθε 2 λεπτά για να μην υπερβούμε το duty cycle του lorawan uplink.

System Workflow

Ο μικροελεγκτής στέλνει λοιπόν MQTT μηνύματα σε μορφή JSON objects στον MQTT Broker που βρίσκεται στο end device, τα μηνύματα αυτά πρέπει να μπορούν να αποθηκεύονται για κάποιο χρονικό διάστημα πάνω στο end device, διότι το μεσοδιάστημα χρόνου κατά το οποίο το end device κάνει τα uplinks μπορεί να αλλάξει αναλόγως της ποσότητας των uplinks και ειδικότερα αν γίνεται χρήση του listen before talk, σε περίπτωση που το κανάλι στο οποίο μεταδίδουμε είναι γεμάτο τότε η συσκευή θα πρέπει να περιμένει πριν προσπαθήσει να κάνει το uplink. Με το σκεπτικό αυτό θα πρέπει έτσι να σχεδιάσουμε ένα απλό FIFO list για την αποθήκευση των MQTT μηνυμάτων απο τον μικροελεγκτή.

Όταν ο μικροελεγκτής στείλει μήνυμα στο end device, το end device θα προσθέσει αυτο το μήνυμα στο FIFO list, όταν έρθει η ώρα για το end device να στείλει το uplink, το πρόγραμμα θα πρέπει να διαβάσει και να αφαιρέσει το τελευταίο διαθέσιμο μήνυμα στην FIFO

list και να το επεξεργαστεί και να σειριοποιήσει το μήνυμα για να μεταδοθεί μέσω του LoRaWAN. Κατά την επεξεργασία και την σειριοποίηση, τα δεδομένα του αισθητήρα εισέρχονται σε μια δομή δεδομένων protobuf. Η δομή αυτή περιέχει όλα τα δεδομένα που διάβασε ο αισθητήρας, όπως θερμοκρασία, υγρασία και άλλα, μαζί περιέχει το unix timestamp της στιγμής που έγινε η σειριοποίηση και αποστολή του μηνύματος με uplink, ένα unix timestamp difference απο την στιγμή της σειριοποίησης και τη στιγμή που έγινε η μέτρηση του χώματος και επίσης το όνομα ή ID του κόμβου με τον αισθητήρα.

Το μήνυμα αφότου φτάσει στον network server και στον application server, μεταφέρεται μέσω MQTT σε έναν backend server γραμμένος σε nodejs, ο server αυτός πρέπει να αποσειριοποιείσει το uplink, να αποθηκεύσει τα δεδομένα σε μια βάση δεδομένων και επίσης, ανάλογα με τα δεδομένα που έρχονται και τις παραμέτρους που θέτονται, ο backend server πρέπει να στείλει τις ανάλογες παραμέτρους για να αλλάξει την ταχύτητα της αντλίας ποτίσματος και την συχνότητα του ποτίσματος. Ο backend server θα πρέπει να γεμίσει μια νέα δομή δεδομένων protobuf χρησιμοποιώντας το ίδιο αρχείο .proto αλλά διαφορετικό μήνυμα σχεδιασμένο μόνο για τις παραμέτρους που χρειάζεται η περισταλτική αντλία. Το μήνυμα αυτό περιλαμβάνει την ταχύτητα περιστροφής της αντλίας, τον χρόνο για τον οποίο θα λειτουργεί η αντλία και τον χρόνο μεσοδιάστημα για τον οποίο η αντλία θα είναι αδρανής.

Ο backend server αφού γεμίσει τη δομή δεδομένων, θα κάνει τη σειριοποίηση και το μήνυμα θα σταλεί στο application server του LoRaWAN δικτύου χρησιμοποιώντας MQTT για να μπεί στο downlink queue και έτσι να σταλεί στο LoRaWAN end device την επόμενη δυνατή στιγμή. Το μήνυμα αφού φτάσει στο end device, θα αποσειριοποιηθεί και το πρόγραμμα που τρέχει στο end device θα βάλει τα δεδομένα σε ένα JSON object το οποίο θα σταλεί στον μικροελεγκτή ο οποίος ελέγχει την περισταλτική αντλία και έπειτα ο μικροελεγκτής θα αλλάξει τις παραμέτρους της αντλίας.

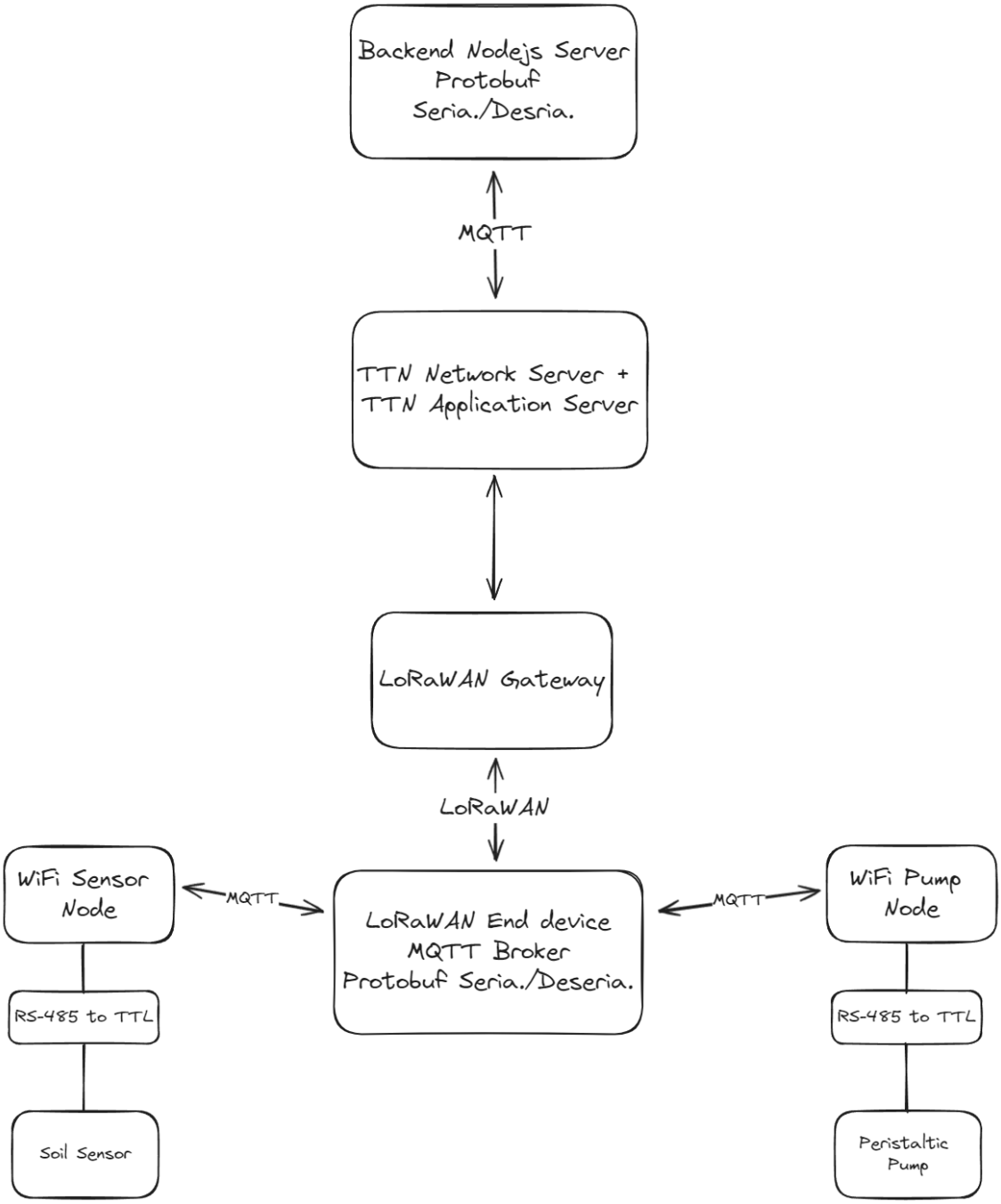


Fig. 8 System Workflow της υλοποίησης μας.

4. Υλοποίηση

4.1. Υλοποίηση hardware

4.1.1. Σχεδιασμός WiFi κόμβου με αισθητήρα εδάφους / περισταλτική αντλία

Η hardware υλοποίηση της εργασίας περιλαμβάνει την ενοποίηση διαφόρων εξαρτημάτων για την δημιουργία ενός ισχυρού συστήματος μετάδοσης και συλλογής δεδομένων από αισθητήρες και τον έλεγχο ενεργοποιητών. Τα κύρια hardware εξαρτήματα περιλαμβάνουν, τον μικροελεγκτή με δυνατότητα WiFi, Real Time Clock, RS-485 to TTL converter, και RS-485 αισθητήρα και RS-485 ενεργοποιητή.

Μικροελεγκτής

Ο μικροελεγκτής της επιλογής μου είναι το ESP-32 διότι υποστηρίζει τεχνολογίες ασύρματης επικοινωνίας όπως WiFi και Bluetooth, έχει αρκετά GPIO pins για τα διάφορα εξαρτήματα που χρειαστούμε, έχει έναν αρκετά ικανό διπύρηνο επεξεργαστή με τους πυρήνες να τρέχουν στα 240 MHz. Προσφέρει άφθονη επεξεργαστική δύναμη για όλα τα tasks που θα χρειαστεί να τρέχουμε πάνω του, όπως tasks για την ασύρματη επικοινωνία, την συλλογή δεδομένων από τον αισθητήρα κλπ. Μας παρέχει επίσης περιφερειακό UART το οποίο θα χρειαστούμε για να συνδέσουμε το RS-485 to TTL converter και περιφερειακό i2c που θα χρειαστεί για το Real Time Clock. Το ESP32 μπορεί να προσφέρει χαμηλή ενεργειακή κατανάλωση με τη χρήση του sleep mode το οποίο το καθιστά πολύ καλή επιλογή για συσκευή με χρήση μπαταρίας ως την κύρια πηγή ενέργειας.

Το software stack του ESP32 είναι δυνατό και με πολλές επιλογές. Υποστηρίζει την χρήση της πλατφόρμας Arduino αλλά μπορεί να γίνει και χρήση της πλατφόρμας ESP IDF που είναι σχεδιασμένη από την Espressif ειδικά για το ESP32 και παρέχει δυνατότητα για χρήση FreeRTOS. Το FreeRTOS κάνει τον προγραμματισμό του μικροελεγκτή αρκετά εύκολο διότι μας δίνει την δυνατότητα να χρησιμοποιήσουμε tasks τα οποία επιτρέπουν την παραλληλοποίηση των λειτουργιών. Και πιο συγκεκριμένα θα χρησιμοποιήσω το ESP32 DEVKIT V1.

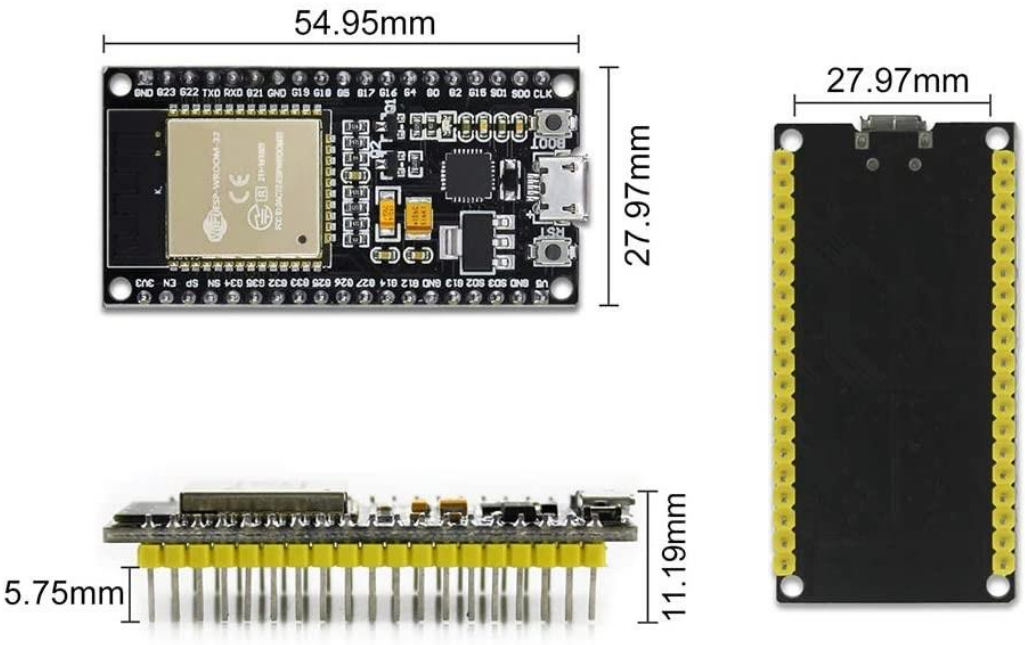


Fig. 9 ESP32 DEVKIT V1 (source: https://grobotronics.com/esp32-development-board-devkit-v1.html)

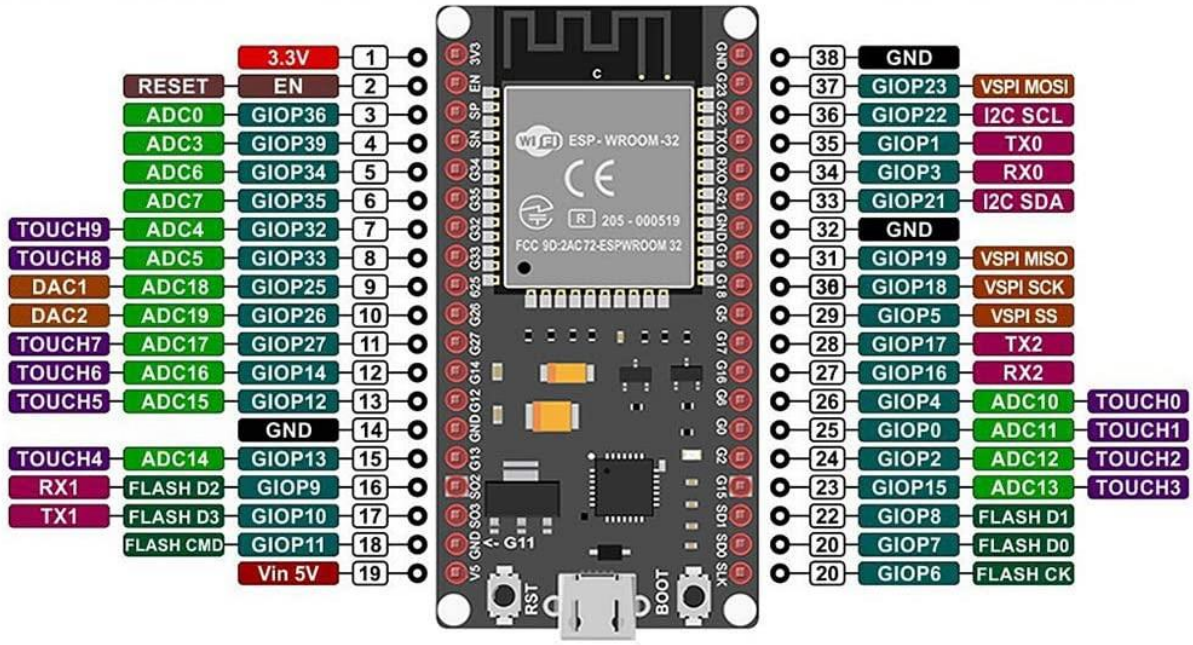


Fig. 10 ESP32 DEVKIT V1 (source: https://grobotronics.com/esp32-development-board-devkit-v1.html)

RS485 Converter

Για την σύνδεση του αισθητήρα και του ενεργοποιητή θα χρειαστούμε κάποιον converter που θα επιτρέψει στον μικροελεγκτή να δημιουργήσει τη διεπαφή που χρειαζόμαστε για την σειριακή επικοινωνία. Η επιλογή για τον converter είναι ένας converter βασισμένος στον πομποδέκτη MAX485 [26]. Ο MAX485 είναι ένας πομποδέκτης με χαμηλή ενεργειακή κατανάλωση που υποστηρίζει RS485 και RS422 σειριακές επικοινωνίες, σχεδιασμένος για την χρήση με βιομηχανικές και εμπορικές εφαρμογές που χρειάζονται αξιοπιστία στη μετάδοση δεδομένων σε μεγάλες αποστάσεις. Μπορεί να φτάσει ταχύτητα δεδομένων μέχρι και 2.5 Mbps, υπάρχει δυνατότητα σύνδεσης μέχρι και 32 πομποδεκτών στον δίαυλο.



Fig. 11 Max485 RS485 to TTL Converter (source: https://www.elecrow.com/uart-ttl-to-rs485-twoway-converter-p-1545.html)

Real Time Clock

Για το real time clock χρησιμοποίησα το Tiny RTC Module DS1307 λόγω του μικρού κόστους και την ευκολία χρήσης του. Επικοινωνεί με τον μικροελεγκτή χρησιμοποιώντας το πρωτόκολλο επικοινωνίας i2c, κρατάει τον χρόνο με ακρίβεια ακόμα και σε διακοπή ρεύματος με την CR2032 μπαταρία που έχει πάνω του.

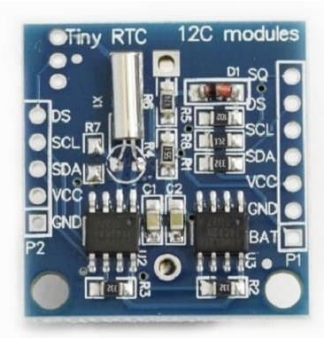


Fig. 12 Tiny RTC Module DS1307 (source: https://www.devobox.com/el/peripheralsen/261-tiny-rtc-module-ds1307-for-arduino.html)

Αισθητήρας και Ενεργοποιητής

Η εργασία χρησιμοποιεί αισθητήρα εδάφους για την παρακολούθηση περιβαλλοντικών παραμέτρων και ενεργοποιητές για έλεγχο. Ο αισθητήρας έχει την δυνατότητα για παρακολούθηση θερμοκρασίας, υγρασίας και αγωγιμότητας του εδάφους και ο ενεργοποιητής

είναι μια περισταλτική αντλία για τον έλεγχο του συστήματος άρδευσης. Και τα δύο χρησιμοποιούν το πρωτόκολλο σειριακής επικοινωνίας RS485 για αξιόπιστη μετάδοση δεδομένων που καλύπτει μεγάλες αποστάσεις.

Πιο συγκεκριμένα για τον αισθητήρα χρησιμοποίησα το Industrial Soil Moisture & Temperature & EC Sensor MODBUS-RTU RS485 (S-Soil MTEC-02A) απο την Seeed Studio. Ο S-Soil MTEC-02A είναι ένας πολυχρηστικός αισθητήρας σχεδιασμένος για μετρήσεις εδάφους με ακρίβεια. Μπορεί να μετρήσει θερμοκρασία, υγρασία και ηλεκτρική αγωγιμότητα του εδάφους, παρέχοντας έτσι σημαντικά δεδομένα για την βελτιστοποίηση γεωργικών και κηπευτικών εφαρμογών.



Fig. 13 3 in 1 Soil Sensor (source: https://solution.seeedstudio.com/product/industrial-soil-moisture-temperature-ec-sensor-modbus-rtu-rs485-s-soil-mtec-02a/)

Για την περισταλτική αντλία επέλεξα μια αντλία της Kamoer μαζί με τον MODBUS driver για τον έλεγχο του βηματικού κινητήρα. Η αντλία είναι σχεδιασμένη για μέτρηση υγρών μεγάλης ακριβείας. Λειτουργεί στα 24v και μπορεί φτάσει μέγιστη ταχύτητα ροής 1800 ml/min και μέγιστη πίεση 2 bar. Μπορεί να περιστραφεί με ταχύτητα μέχρι και 500 RPM, καθιστώντας την κατάλληλη για εφαρμογές που απαιτούν σταθερή και ακριβής διανομή υγρών. Ο βηματικός κινητήρας εξασφαλίζει τον ακριβής έλεγχο της ταχύτητας ροής που την κάνει ιδανική για χρήση σε βιομηχανικά και εργαστηριακά περιβάλλοντα που χρειάζονται ακρίβεια στην μεταχείριση υγρών.



Fig. 14 Peristaltic Pump (source:https://www.ebay.com/itm/Kamoer-High-Flow-Peristaltic-Pump-with-Stepper-Motor-OEM-Metering-Liquid-Water-/352819069395)



Fig. 15 Pump Motor MODBUS Driver (source: https://vi.aliexpress.com/item/32897655207.html)

4.1.2. Σχεδιασμός LoRaWAN End Device και MQTT Broker

Για το edge device που θα πάρει τον ρόλο του LoRaWAN end device και του MQTT broker επέλεξα το Raspberry Pi 3 λόγω της απόδοσης που προσφέρει με σχετικά χαμηλό κόστος. Θα λειτουργεί ως η συσκευή που επικοινωνεί με το gateway της TTN χρησιμοποιώντας LoRaWAN αλλά και με τους WiFi κόμβους συλλέγοντας τα δεδομένα των αισθητήρων και στέλνοντας τις εντολές για τις αλλαγές των παραμέτρων της αντλίας

άρδευσης. Σε συνδυασμό με το Dragino Pi HAT που παρέχει τον απαραίτητο πομποδέκτη για επικοινωνία LoRaWAN. Το Dragino Pi HAT χρησιμοποιεί τον πομποδέκτη SX1276 για LoRa, επιτρέποντας του την επικοινωνία χαμηλής ενεργειακής κατανάλωσης. Η εγκατάσταση του hardware περιλαμβάνει την σύνδεση του Pi HAT πάνω στα GPIO pins του Raspberry Pi και έπειτα την σύνδεση ενός RTC επίσης πάνω στα GPIO του Raspberry Pi.



Fig. 16 Raspberry Pi 3 Model B (source: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/)

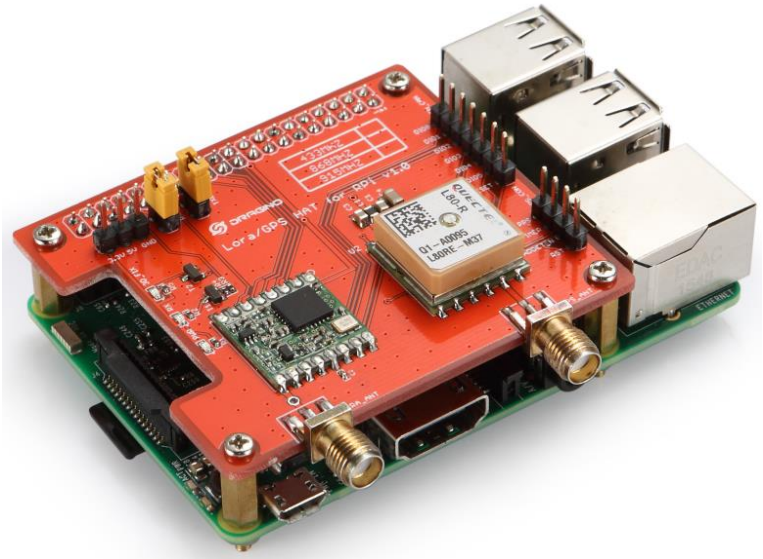
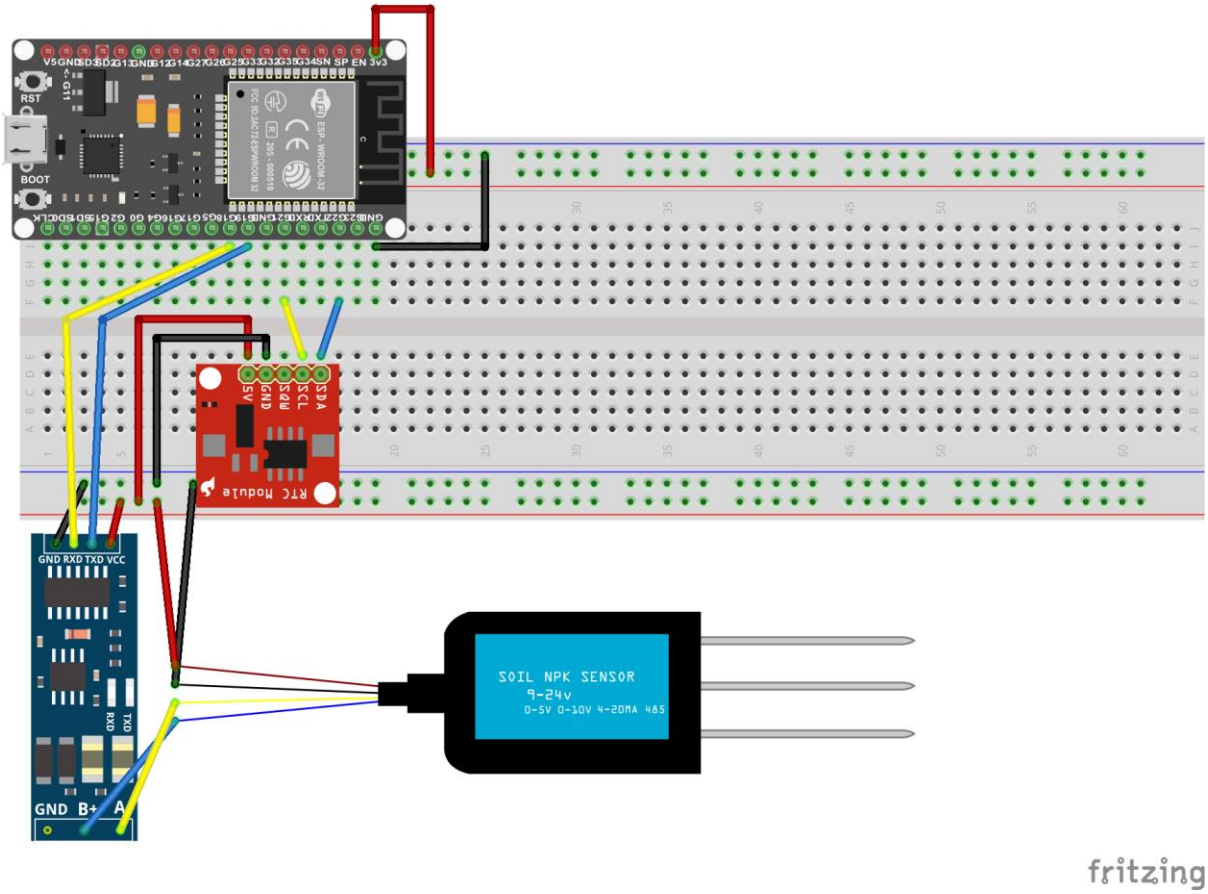


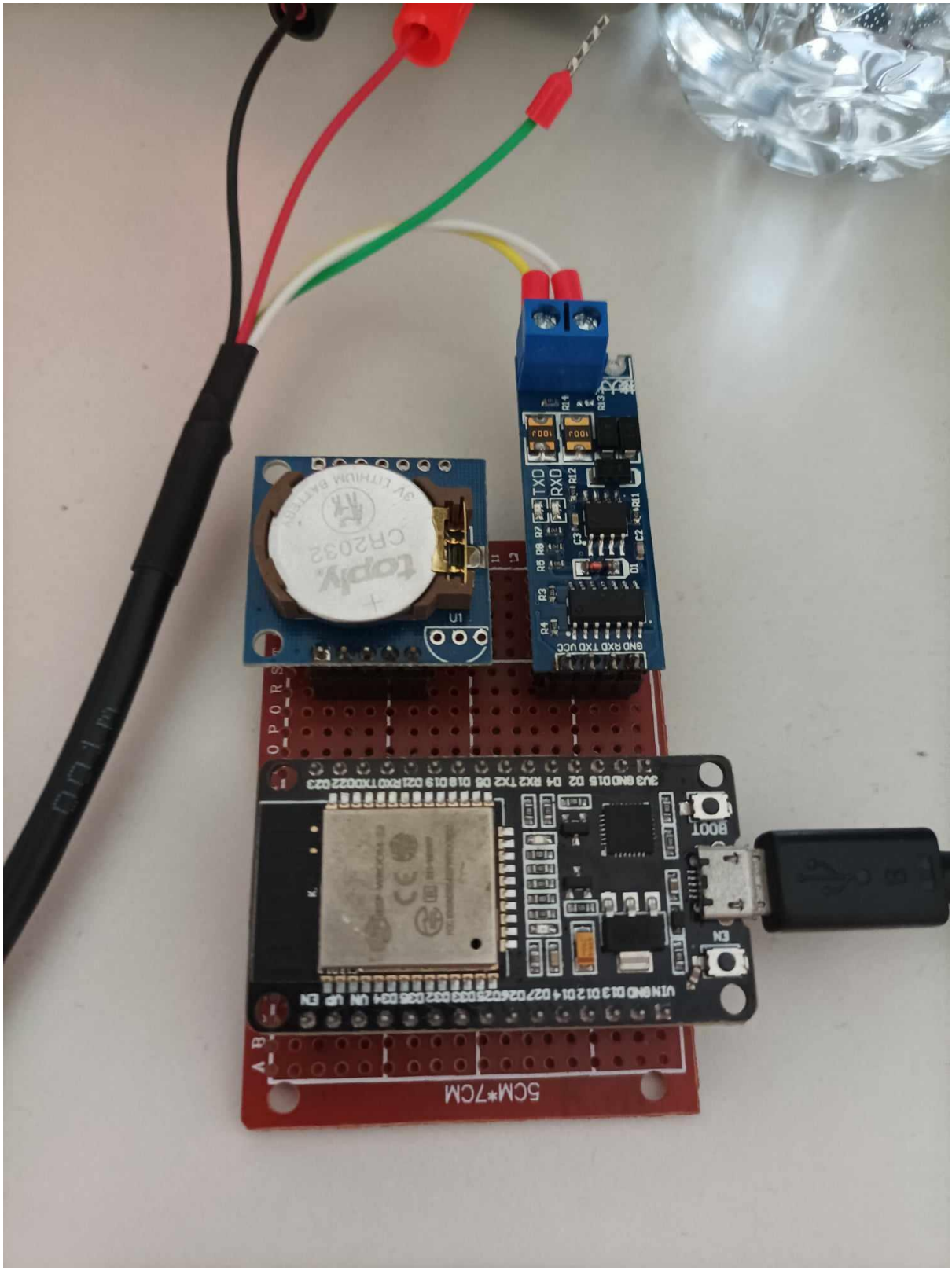
Fig. 17 Rpi 3 With Dragino Pi HAT (source: https://www.dragino.com/products/lora/item/106-lora-gps-hat.html)

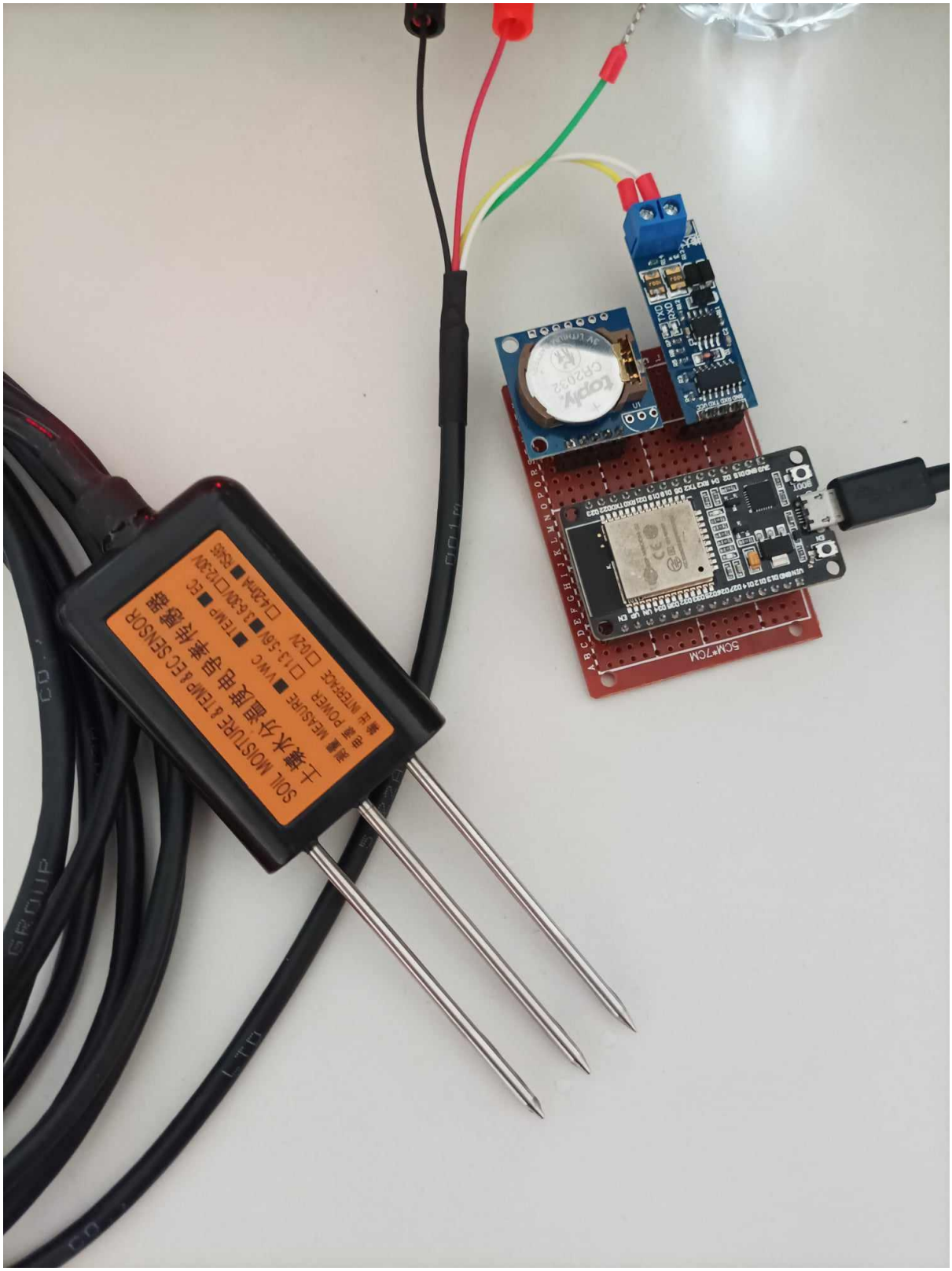
Οι 2 κόμβοι έχουν παρόμοια συνδεσμολογία με τα εξαρτήματα τους, η μόνη ουσιώδης διαφορά είναι η συσκευή RS-485 με την οποία επικοινωνεί ο μικροελεγκτής. Στην παρακάτω εικόνα βλέπουμε τις συνδέσεις του μικροελεγκτή με τα εξαρτήματα του. Το DS1307 RTC

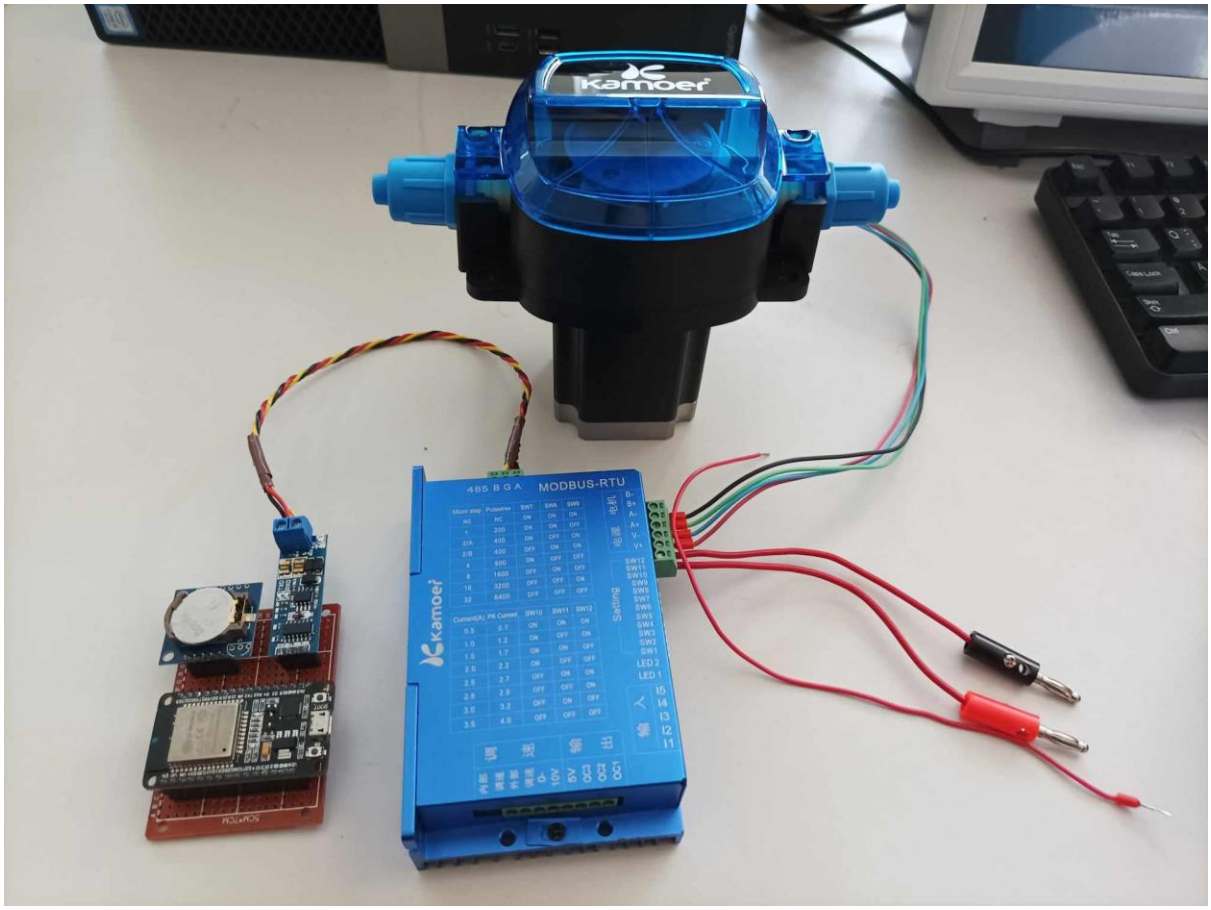
συνδέεται στα pins 21 και 22 του μικροελεγκτή και επικοινωνεί με I2C. Ο RS-485 to TTL converter συνδέεται στα pins 18 και 19 του μικροελεγκτή και επικοινωνεί με UART.



Στην παραπάνω εικόνα αποδεικνύεται η συνδεσμολογία του μικροελεγκτή με τον αισθητήρα εδάφους, ο κόμβος που φέρει την αντλία ποτίσματος έχει ακριβώς την ίδια συνδεσμολογία με την μόνη διαφορά ότι στα άκρα B+ και A- του RS-485 συνδέονται τα καλώδια του driver του βηματικού κινητήρα της αντλίας.


TXD
RXD
GND RXD TXD VCC
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
BOOT
EN
5CM*7CM


SOIL MOISTURE & TEMP & EC SENSOR
土壤水分温度电导率传感器
测量 MEASURE ■ VWC ■ TEMP ■ EC
电源 POWER □3.3-5V ■3.6-30V □12-30V
输出 INTERFACE □0-2V ■RS485
SCM*7CM



4.2. Υλοποίηση Software

Οι μικροελεγκτές είναι αρκετά περιορισμένοι από άποψη επεξεργαστικής δύναμης και μνήμης, πράγμα που σημαίνει ότι είναι αρκετά δύσκολοι στον προγραμματισμό και ο κώδικας που γράφουμε για αυτούς πρέπει να είναι προσεγμένος και να μην έχει άχρηστες και περιττές. Ο κώδικας μας θα πρέπει να είναι εύκολος στο διάβασμα, την κατανόηση και στην περαιτέρω ανάπτυξη του. Ο κώδικας πρέπει να είναι αποτελεσματικός στην λειτουργία για την οποία έχει σχεδιαστεί να φέρει εις πέρας.

Η πλατφόρμα που επέλεξα για να προγραμματίσω τους μικροελεγκτές ESP32 είναι η πλατφόρμα ESP IDF και την προσφέρει η εταιρεία που σχεδίασε και τον μικροελεγκτή της επιλογής μας. Είναι διαθέσιμη για ολα τα μεγάλα λειτουργικά συστήματα όπως Windows, Linux και Mac OS. Χρησιμοποιώντας το ESP IDF κάποιος μπορεί να προγραμματίσει οποιαδήποτε παραλλαγή του ESP32 χρησιμοποιώντας τις γλώσσες προγραμματισμού C και C++. Το ESP IDF προσφέρει έτοιμο κώδικα και βιβλιοθήκες για την χρήση όλων των περιφερειακών συστημάτων και χαρακτηριστικών που προσφέρουν οι μικροελεγκτές, όπως για παράδειγμα τον έλεγχο των GPIO, τον έλεγχο του WiFi και του bluetooth κλπ.

Η πλατφόρμα προσφέρει επίσης λειτουργία επέκτασης με το ESP IDF Registry, έτσι επιτρέποντας στην κοινότητα να δημιουργήσει και να δημοσιεύσει βιβλιοθήκες κώδικα για χαρακτηριστικά τα οποία ίσως να μην υπάρχουν out of the box απο την espressif. Η πλατφόρμα είναι αρκετά εύκολη στην εγκατάσταση της, υπάρχουν online οδηγίες και manuals για την εγκατάσταση και την χρήση της μαζί με documentation για όλα τα κομμάτια κώδικα. Στην δική μου περίπτωση η εγκατάσταση της πλατφόρμας στα Linux περιλαμβάνει το download

ενός GitHub Repo και το execution του bash script που έρχεται με το Repo. Έπειτα ο χειρισμός της πλατφόρμας γίνεται μέσω του τερματικού και ο καθένας μπορεί να γράψει κώδικα με τον επεξεργαστή κειμένου της επιλογής του.

4.2.1. WiFi Κόμβος αισθητήρα

4.2.1.1. Inclusions και Declarations

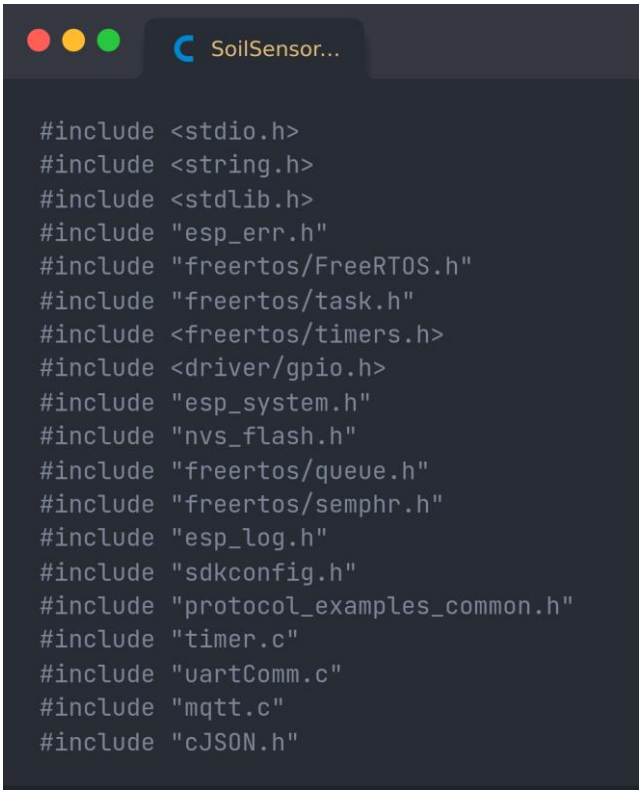


Fig. 18 Includes

Το πρώτο πράγμα που βλέπουμε στον κώδικα είναι το command #include<library.h>, αυτή η εντολή περιλαμβάνουμε τις βιβλιοθήκες που χρειαζόμαστε για το project μας.

- Stdio.h, stdlib.h: Είναι βασικές βιβλιοθήκες της C για input output στο τερματικό και για βασικές λειτουργίες όπως management της μνήμης.
- string.h : Είναι βιβλιοθήκη για χειρισμό text.
- esp_err.h : Είναι βιβλιοθήκη για τον χειρισμό σφαλμάτων.
- freertos/…h : Ολα τα include του freertos είναι κομμάτια του real time operating system που θα τρέχει πάνω στο ESP32, συμπεριλαμβανομένων tasks, timers, schedulers, semaphores, queues και άλλα.
- Esp_system.h : το σύστημα του ESP32.
- Nvs_flash.h : Βιβλιοθήκη για την αποθήκευση μεταβλητών στην μνήμη όπως το SSID και ο κωδικός του WiFi Access point στο οποίο θα συνδέεται το ESP32.

- Protocol_examples_common.h : Κομμάτια boilerplate κώδικα για την σύνδεση στο WiFi.
- Timer.c : Είναι το αρχείο που ελέγχει το RTC
- uartComm.c : Είναι το αρχείο κώδικα που ελέγχει την επικοινωνία UART με τον RS485 to TTL converter και επίσης την ανάγνωση των μεταβλητών απο τον αισθητήρα εδάφους.
- Mqtt.c : Είναι το αρχείο κώδικα για την επικοινωνία MQTT με το edge device
- cJSON.h : Είναι μια βιβλιοθήκη που δίνει δυνατότητες χειρισμού JSON objects.

4.2.1.2. Main Function

SoilSensor...

```c
void app_main(void)
{
    // INIT WIFI
    ESP_ERROR_CHECK(nvs_flash_init());
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    ESP_ERROR_CHECK_WITHOUT_ABORT(example_connect());

    // Task for MQTT Communication
    xTaskCreate(mqtt_main, "mqtt_task", 4096, NULL, 10, &mqttTaskHandle);
    // Task for the UART communication;
    xTaskCreate(echo_task, "uart_echo_task", ECHO_TASK_STACK_SIZE, NULL, ECHO_TASK_PRIO, &uartTaskHandle);

    // I2C Connection for RTC
    ESP_ERROR_CHECK_WITHOUT_ABORT(i2cdev_init());
    // Task for RTC
    // setTime();
    xTaskCreate(ds1307_test, "ds1307_test", configMINIMAL_STACK_SIZE * 8, NULL, 5, &timerTaskHandle);

    double *uartvalues;
```

Στις γραμμές 29 μεχρι και 32 ξεκινάμε το WiFi. Παρακάτω κάνουμε init τα tasks για την επικοινωνία MQTT, την επικοινωνία UART για τον αισθητήρα, έπειτα κάνουμε init την επικοινωνία i2c για χρήση με το RTC και μετά το task για τον έλεγχο του RTC και τελευταία ξεκινάμε μια double μεταβλητή δείκτη.

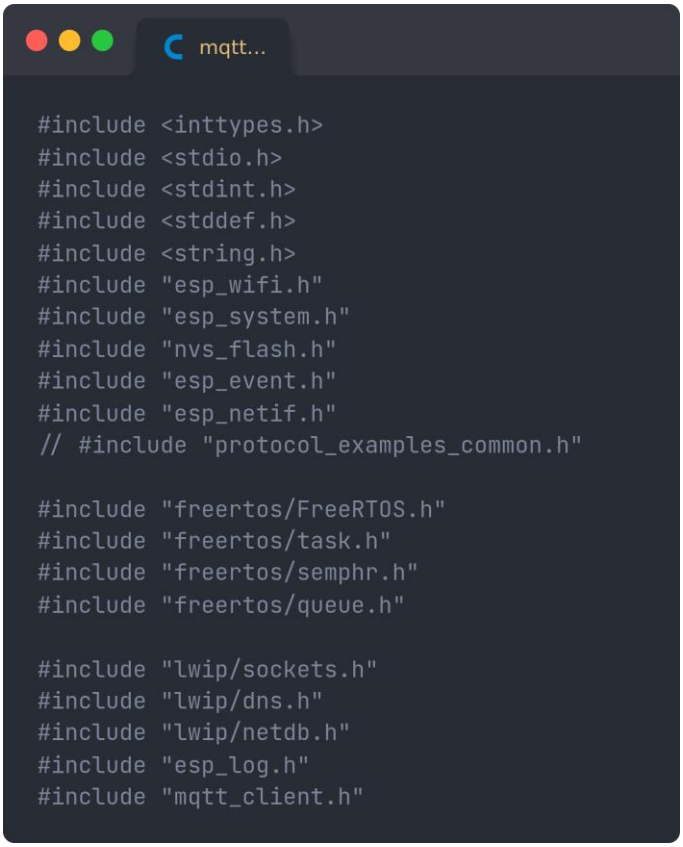
4.2.1.3. Main While



Στην while του main προγράμματος τρέχουμε κομμάτια του κώδικα σε λούπα για όσο ο μικροελεγκτής είναι στο ρεύμα. Στην αρχή κάθε επανάληψης καλούμε μια συνάρτηση από το αρχείο uartComm.c το οποίο μας επιστρέφει τις μετρήσεις απο τον αισθητήρα εδάφους. Στην συνέχεια δημιουργούμε 2 JSON objects, το ένα είναι το γενικό μήνυμα και το δεύτερο θα περιέχει τις μετρήσεις του αισθητήρα. Στο αρχικό object βάζουμε τον ρόλο του του κόμβου, την χρονική στιγμή που έγινε η μέτρηση καλώντας μια συνάρτηση από το αρχείο timer.c. Θέτουμε το id της συσκευής, βάζουμε τις μετρήσεις του αισθητήρα στο object των δεδομένων και μετά βάζουμε το ένα object μέσα στο άλλο. Έπειτα κάνουμε φορματ το json object και το κάνουμε publish καλώντας μια συνάρτηση από το αρχείο mqtt.c και τέλος κάνουμε free την μνήμη του json object και κάνουμε delay 2 λεπτών πριν ξανα γίνει η επανάληψη της while.

4.2.1.4. MQTT Task

Inclusions



Τα inclusions στο mqtt αρχείο είναι πάνω κάτω τα ίδια με μερικές προσθήκες.

- lwip/….h : Ολα τα includes από το Lightweight IP stack είναι βιβλιοθήκες που αφορούν δικτυακά functions. Πιο συγκεκριμένα για τα berkeley sockets, για Domain Name System resolution και για network database operations.
- Mqtt_client.h : Είναι βιβλιοθήκη που παρέχει λειτουργία και functions για την εφαρμογή MQTT client.

Ο κώδικας του mqtt.c είναι κυρίως boilerplate κώδικας που έρχεται έτοιμος. Ο κώδικας περιέχει το handler function για ολα τα events που μπορεί να συμβούν σε έναν mqtt client, όπως για παράδειγμα

- MQTT_EVENT_CONNECTED : κάνει publish κάποιο μήνυμα της επιλογής μας κατά την σύνδεση του client στον broker
- MQTT_EVENT_DISCONNECTED : κάνει publish κάποιο μήνυμα της επιλογής μας κατά την αποσύνδεση του client στον broker
- MQTT_EVENT_PUBLISHED : δημιουργεί event όταν έρθει κάποιο μήνυμα σε ένα συγκεκριμένο topic
- MQTT_EVENT_DATA : εμφανίζει τα δεδομένα που στάλθηκαν

4.2.1.4.1. sendMessage

Το κυριότερο function που μας ενδιαφέρει απο το συγκεκριμένο αρχείο είναι το function που δέχεται ένα topic και ένα μήνυμα όταν καλείται και κάνει publish το συγκεκριμένο μήνυμα στο συγκεκριμένο topic.

mqtt...

```
void sendMessage(char *topic, char *msg) {
    int checkSuccess = esp_mqtt_client_publish(client, topic, msg, 0, 0, 0);
}
```

4.2.2. UART Task

mqtt...

```
uint8_t values[16];

// Variables to hold the values
double temp;
double vowc;
double elco;

uint16_t t;
uint16_t v;
uint16_t e;

static void echo_task(void *arg)
{
    uart_config_t uart_config = {
        .baud_rate = BAUD_RATE,
        .data_bits = UART_DATA_8_BITS,
        .parity = UART_PARITY_DISABLE,
        .stop_bits = UART_STOP_BITS_1,
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
        .rx_flow_ctrl_thresh = 122,
        .source_clk = UART_SCLK_DEFAULT,
    };

    esp_log_level_set(UARTAG, ESP_LOG_INFO);

    ESP_LOGI(UARTAG, "Start RS485 application test and configure UART.");

    // Install UART driver (we don't need an event queue here)
    // In this example we don't even use a buffer for sending data.
    ESP_ERROR_CHECK(uart_driver_install(uart_num, BUF_SIZE * 2, 0, 0, NULL, 0));

    // Configure UART parameters
    ESP_ERROR_CHECK(uart_param_config(uart_num, &uart_config));

    ESP_LOGI(UARTAG, "UART set pins, mode and install driver.");

    // Set UART pins as per KConfig settings
    ESP_ERROR_CHECK(uart_set_pin(uart_num, ECHO_TEST_TXD, ECHO_TEST_RXD, ECHO_TEST_RTS, ECHO_TEST_CTS));

    // Set RS485 half duplex mode
    ESP_ERROR_CHECK(uart_set_mode(uart_num, UART_MODE_RS485_HALF_DUPLEX));

    // Set read timeout of UART TOUT feature
    ESP_ERROR_CHECK(uart_set_rx_timeout(uart_num, ECHO_READ_TOUT));
```

Στο αρχείο uartComm.c τα inclusions είναι επίσης παρόμοια με τα υπόλοιπα με επιπρόσθετο το driver/uart.h που είναι η βιβλιοθήκη για χρήση του uart. Δηλώνουμε 1 πίνακα μεταβλητών και 6 μεταβλητές που θα κρατάνε τις τιμές από τις μετρήσεις του αισθητήρα. Το main function του uartComm.c αρχικοποιεί τις μεταβλητές που χρειάζονται για την λειτουργία του UART.

Θέτονται κάποιες παράμετροι από το sdkconfig όπως η ταχύτητα επικοινωνίας (baud rate, τιμή 9600) μαζί με κάποιες άλλες. Έπειτα γίνεται εγκατάσταση του UART driver και θέτονται τα pins. Το πιο σημαντικό function του αρχείου είναι το getSoilData().

4.2.2.1. GetSoilData

```c
double* getSoilData() {
    static double values1[3];
    char message[] = {0x01, 0x04, 0x0, 0x0, 0x0, 0x03, 0xB0, 0x0B};
    echo_send(uart_num, message, 8);
    int len;
    // Read data from UART
    len = uart_read_bytes(uart_num, data, BUF_SIZE, PACKET_READ_TICS);

    if (len > 0)
    {
        ESP_LOGI(UARTAG, "Received %u bytes:", len);
        printf("[ ");
        for (int i = 0; i < len; i++)
        {
            values[i] = (uint8_t)data[i];
            printf("0x%.2X ", (uint8_t)data[i]);
        }
        printf("] \n");
    }

    ESP_ERROR_CHECK(uart_wait_tx_done(uart_num, 10));
    t = values[3] << 8 | values[4];
    v = values[5] << 8 | values[6];
    e = values[7] << 8 | values[8];
    values1[0] = (double)t / 100.0;
    values1[1] = (double)v / 100.0;
    values1[2] = (double)e;
    return values1;
}
```

Για αρχή δημιουργούμε έναν πίνακα double 3 θέσεων, και την αρχικοποίηση του μηνύματος που θα στείλουμε μέσω UART στον RS485 αισθητήρα εδάφους. Το μήνυμα πρέπει να είναι σε μορφή δεκαεξαδικών αριθμών. Πιο συγκεκριμένα η πρώτη τιμή είναι η διεύθυνση του αισθητήρα, η επόμενη τιμή είναι το function που θέλουμε να πραγματοποιήσει ο αισθητήρας και σε αυτή τη περίπτωση το function είναι να διαβάσει κάποιον ή κάποιους input registers, η τρίτη τιμή είναι η αρχή της διεύθυνσης που θέλουμε να διαβάσουμε, η τέταρτη τιμή είναι ο αριθμός του register που θέλουμε να διαβάσουμε και οι τελευταίοι δύο τιμές είναι το CRC check της UART επικοινωνίας. Το μήνυμα μετά στέλνεται με χρήση του function echo_send() που δέχεται ως παραμέτρους το UART port που έχει δοθεί στο sdkconfig, το μήνυμα που θέλουμε να στείλουμε και το μέγεθος του μηνύματος.

Στην συνέχεια το πρόγραμμα διαβάζει την απάντηση από τον αισθητήρα χαρακτήρα προς χαρακτήρα και τους αποθηκεύει στον πίνακα μεταβλητών που δηλώσαμε στην αρχή του προγράμματος. Έπειτα το πρόγραμμα για να μετατρέψει το μήνυμα που έλαβε σε τιμές που μπορούμε να χρησιμοποιήσουμε πρέπει να κάνει bit shift αριστερά κατ'α 8 bit το most significant byte και συνδυάζεται με το least significant byte χρησιμοποιώντας έναν bitwise OR operator. Και τέλος γίνονται 2 διαιρέσεις για στις δύο πρώτες μεταβλητές και οι μεταβλητές επιστρέφονται στην αρχική κλήση της συνάρτησης.

Echo_send

Το function που στέλνει τα μηνύματα στον αισθητήρα δέχεται ως παραμέτρους το port του uart, το μήνυμα που θέλουμε να στείλουμε και το μέγεθος του μηνύματος. Το function uart_write_bytes επιστρέφει έναν int που πρέπει να είναι ίσο με το μέγεθος με το μήνυμα που

στείλαμε, αν αυτός ο αριθμός είναι διαφορετικός υποθέτουμε ότι υπήρξε σφάλμα και ότι το μήνυμα δεν στάλθηκε σωστά.

```c
static void echo_send(const int port, const char *str, uint8_t length)
{
    if (uart_write_bytes(port, str, length) != length)
    {
        ESP_LOGE(UARTAG, "Send data critical failure.");
        abort();
    }
}
```

4.2.3. Real Time Clock Task

getRTCEpoch

Το πρόγραμμα στο αρχείο του RTC έχει μόνο μια επιπρόσθετη βιβλιοθήκη την time.h που μας επιτρέπει να χρησιμοποιήσουμε και να χειριστούμε τιμές χρόνου. Το κύριο και πιο σημαντικό function είναι το getRTCEpoch(). Το function αυτό επιστρέφει την τρέχουσα τιμή χρόνου σε μορφη unix timestamp στο σημείο που έγινε η κλήση του function.

```c
time_t getRTCEpoch(){
    struct tm newTime;
    ds1307_get_time(&dev, &newTime);
    // Create epoch from timestamp
    time_t unix_time = mktime(&newTime);

    // ESP_LOGI(TIMERTAG, "Time is: %lld", unix_time);
    return unix_time;
}
```

4.2.4. WiFi Κόμβος Αντλίας

Ο κόμβος που φέρει τον ενεργοποιητή ή στην περίπτωση μας την περισταλτική αντλία τρέχει πρακτικά τον ίδιο κώδικα με μερικές διαφορές στα αρχεία του UART του MQTT client και μερικές προσθήκες στο main πρόγραμμα του. Τα inclusions παραμένουν τα ίδια.

4.2.4.1. Main task

Στο main πρόγραμμα έγινε η προσθήκη ενός queue και ενός νέου task το οποίο πρακτικά ελέγχει το πότισμα. Το queue θα μας χρειαστεί για να μπορούμε να μεταφέρουμε τα MQTT μηνύματα που δέχεται ο MQTT client στο UART κομμάτι του προγράμματος και έτσι να αλλάξουν οι παράμετροι για το πότισμα. Παρακάτω θα δούμε και τον κώδικα του νέου task

main_app...

```
// Queue
QueueHandle_t pumpConfigQueue;
pumpConfigQueue = xQueueCreate(5, sizeof(cJSON *));

// Pump control task
xTaskCreate(movePump, "pump stack", configMINIMAL_STACK_SIZE * 8, NULL, 5, &movePumpHandle);
```

4.2.4.2. MQTT Task

mqtt...

```
char *read_topic;
char *read_data;

read_topic = (char*)malloc((event->topic_len + 1) *sizeof(char));
read_data = (char*)malloc((event->data_len + 1) *sizeof(char));

// ESP_LOGI(MQTTTAG, "Topic is %.*s", event->topic_len, event->topic);

cJSON *root;

sprintf(read_topic, "%.*s", event->topic_len, event->topic);
sprintf(read_data, "%.*s", event->data_len, event->data);
if (strcmp(read_topic,"/GH_1/pump") == 0) {
    root = cJSON_Parse(read_data);
    if (root == NULL) {
        ESP_LOGI(MQTTTAG, "JSON obj is empty\n");
        cJSON_Delete(root);
    }

    // ESP_LOGI(MQTTTAG, "This is my JSON\n%s",cJSON_Print(root));
    // ESP_LOGI(MQTTTAG, "pump config sent");

    // check for space in queue and send
    if (uxQueueSpacesAvailable(pumpConfigQueue) > 0) {
        // Send data to the message queue
        if (xQueueSend(pumpConfigQueue, &root, 0) != pdPASS) {
            printf("Failed to send data to the queue.\n");
        }
    } else {
        printf("Queue is full. Cannot send data.\n");
    }
}

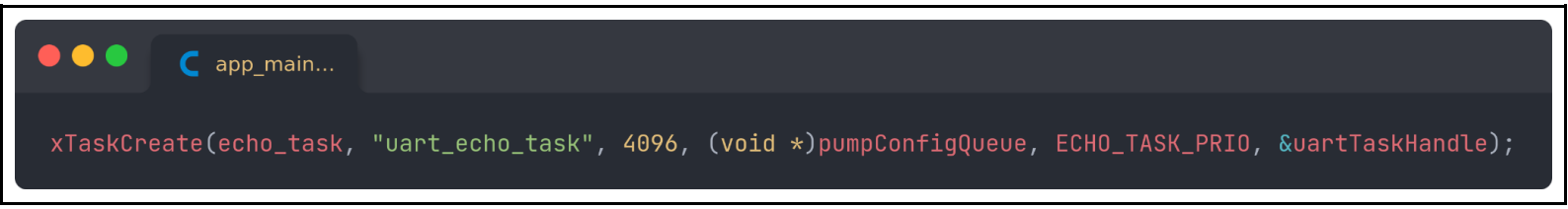
free(read_topic);
free(read_data);
```

Στο mqtt client προστέθηκε αυτό το κομμάτι κώδικα. Δηλώνουμε δύο δείκτες char για το topic και τα δεδομένα του mqtt μηνύματος που θα δεχτούμε. Κάνουμε malloc και cast τους δείκτες. Δημιουργούμε ένα json object για να μπορέσουμε να κάνουμε parse το μήνυμα που δεχτήκαμε. Έπειτα χρησιμοποιούμε το function sprintf για να κάνουμε format το τους χαρακτήρες για το topic και τα δεδομένα του μηνύματος που δεχτήκαμε. Μετά από το φορματ

γίνεται έλεγχος για να σιγουρευτούμε ότι το μήνυμα είναι για εμάς τσεκάροντας αν το topic του μηνύματος είναι το /GH_1/pump. Αν ο έλεγχος περάσει τότε κάνουμε parse το μήνυμα από το mqtt και τσεκάρουμε αν το parsing έγινε σωστά. Αν υπάρξει κάποιο σφάλμα τότε διαγράφουμε την μεταβλητή json που δημιουργήσαμε. Αν όλα πήγαν καλά τότε είναι η ώρα να βάλουμε το μήνυμα στο queue που αρχικοποιήσαμε στο main πρόγραμμα. Πρώτα ελέγχουμε αν το queue έχει χώρο, αν υπάρχει όντως χώρος τότε βάζουμε το json object στο queue αλλιώς εμφανίζουμε ένα μήνυμα στην κονσόλα. Αφότου τελειώσει όλες οι λειτουργίες που έχουμε ελευθερώνουμε τις μεταβλητές του topic και των δεδομένων του μηνύματος.

4.2.4.3. UART Task και Move Pump Task

Μια κύρια αλλαγή στο UART task είναι η αποστολή του queue list ως argument που γίνεται στην δήλωση του task στο main πρόγραμμα.



uartComm...

```c
QueueHandle_t pumpConfigQueue;
pumpConfigQueue = (QueueHandle_t)arg;
cJSON *test;
setPumpSpeed(5);
while (1) {

    // Queue receive if to check cJSON object
    if (xQueueReceive(pumpConfigQueue, &test, 0) == pdTRUE) {
        // Process received data
        if (test == NULL) {
            ESP_LOGI(UARTAG, "Received cJSON object is NULL.\n");

            cJSON_free(test);
        } else {
            ESP_LOGI(UARTAG, "Received data from task: %s \n", cJSON_Print(test));

            cJSON *cmd = cJSON_GetObjectItemCaseSensitive(test, "cmd");
            cJSON *deviceId = cJSON_GetObjectItemCaseSensitive(test, "deviceId");
            cJSON *timeInterval = cJSON_GetObjectItemCaseSensitive(test, "interval");
            cJSON *timeDuration = cJSON_GetObjectItemCaseSensitive(test, "duration");
            cJSON *speed = cJSON_GetObjectItemCaseSensitive(test, "speed");

            if (cJSON_IsString(deviceId)) {

                ESP_LOGI(UARTAG, "we are here");
                interval = timeInterval->valuedouble;
                duration = timeDuration->valuedouble;
                setPumpSpeed(speed->valueint);
            } else {
                ESP_LOGI(UARTAG, "JSON field is not of expected type");
            }
        }
    }
    vTaskDelay(pdMS_TO_TICKS(1000));
}
```

Αυτό το κομμάτι κώδικα είναι προσθήκη στο echo_task, στο αρχείο uartComm.c. Στην αρχή κάνουμε δήλωση μεταβλητών για να μπορούμε να χειριστούμε το queue που δεχόμαστε ως argument απο το κάλεσμα του task στο main πρόγραμμα και για να μπορούμε να χειριστούμε το json object που θα διαβάσουμε μέσα από το queue. Αφού μπούμε στο while loop, γίνεται έλεγχος για το αν υπάρχει μήνυμα στο queue, έπειτα γίνεται έλεγχος για το αν το μήνυμα έγκυρο json object. Αν το μήνυμα δεν είναι έγκυρο json object τότε κάνουμε free την

μεταβλητή που δηλωσαμε. Αν όλα πάνε καλά κάνουμε parse το object και κρατάμε τις παραμέτρους, τσεκάρουμε αν το deviceId είναι όντως text τότε περνάμε τις παραμέτρους που χρειαζόμαστε για να ρυθμίσουμε το πότισμα σε μεταβλητές global που έχουμε δηλώσει την αρχή του προγράμματος αμέσως μετά τα inclusions και επίσης γίνεται κλήση του function setPumpSpeed() το οποίο θέτει την ταχύτητα με την οποία θα γυρνάει.

MovePump Task

uartComm...

```
static void movePump(void *pvParameters){
    ESP_LOGI(UARTAG, "hello");
    while (1) {
        forwardGo();
        vTaskDelay(pdMS_TO_TICKS((1000 * 60)* duration));
        forwardStop();
        ESP_LOGI(UARTAG, "This many times %d", counter++);
        // vTaskDelay(pdMS_TO_TICKS((1000 * 60)* interval));
        vTaskDelay(pdMS_TO_TICKS((1000 * 60)* interval));
    }
}
```

Το movePump task χειρίζεται το πότισμα με την αντλία και πιο συγκεκριμένα, το χρονικό διάστημα για το οποίο γυρνάει η αντλία και τον μεσοδιάστημα για το οποίο η αντλία μένει ακίνητη. Μέσα στο task υπάρχει μια while loop με τις εντολές που χρειαζόμαστε για τις κινήσεις της αντλίας και τα delays για το χρονικό διάστημα και το μεσοδιάστημα που ανέφερα. Υπάρχουν επίσης 2 κλήσεις function με ονόματα forwardGo() και forwardStop(). Αυτά τα functions στέλνουν τις αντιστοιχες εντολές μέσω UART στον driver του βηματικού κινητήρα. Παρακάτω είναι τα functions για τις κινήσεις της αντλίας και την ρύθμιση της ταχύτητας.

Κάθε function στέλνει ένα συγκεκριμένο μήνυμα, ένα array απο δεκαεξαδικές τιμές, η μόνη διαφορά με το function για τον έλεγχο της ταχύτητας είναι ότι δέχεται ώς παράμετρο έναν ακέραιο αριθμό τον οποίο πρέπει να τον φορτώσουμε σε συγκεκριμένη θέση στο μήνυμα. Η δομή των μηνυμάτων της αντλίας έχουν ως εξής. Για την ρύθμιση της ταχύτητας: Το πρώτο μονό byte είναι ο αριθμός της συσκευής, το δεύτερο μονό byte είναι το id του function που θέλουμε να κάνει η αντλία, τα επόμενα 2 bytes είναι η αρχή της διεύθυνσης, τα επόμενα 2 bytes είναι ο αριθμός των registered αριθμών, το έβδομο byte είναι ο αριθμός των data bytes, τα επόμενα δύο bytes είναι 00 για το πρωτο και το δεύτερο θα είναι η ταχύτητα της αντλίας που έχουμε δεχτεί ως παράμετρο από το function με μέγιστο αριθμό το 255 και τα τελευταία δύο bytes το CRC check του UART.

Οι εντολές για εμπρός, εμπρός αλτ, πίσω και πίσω αλτ είναι αρκετά όμοια. Για αρχή έχουν στη θέση του πρώτου byte τον αριθμό της συσκευής, μετά το function που θέλουμε να κάνει ο κινητήρας, το τρίτο και τέταρτο byte είναι η διεύθυνση του πηνίου που θέλουμε να ελέγξουμε δηλαδή αν θέλουμε τον κινητήρα να πάει εμπρός ή πίσω. Τα επόμενα δύο bytes είναι αν θέλουμε να ξεκινήσει την κίνηση η αντλία ή να σταματήσει την κίνηση. 00 00 είναι η

εντολή του αλτ και FF 00 είναι η εντολή για κίνηση. Τα byte 00 04 είναι το coil address για εμπρός και το 00 05 το coil address για πίσω.

uartComm...

```
void forwardGo(){
    ESP_LOGI(UARTAG, "Going forward");
    // int forward[] = {0x01, 0x05, 0x0, 0x04, 0xFF, 0x0, 0xCD, 0xFB};
    char forward[] = {0x01, 0x05, 0x0, 0x04, 0xFF, 0x0, 0xCD, 0xFB};
    echo_send(uart_num, forward, sizeof(forward));
}

void forwardStop(){
    ESP_LOGI(UARTAG, "Stopping");
    // int forward_stop[] = {0x01, 0x05, 0x0, 0x04, 0x0, 0x0, 0x8C, 0x0B};
    char forward_stop[] = {0x01, 0x05, 0x0, 0x04, 0x0, 0x0, 0x8C, 0x0B};
    echo_send(uart_num, forward_stop, sizeof(forward_stop));
}

void reverseGo(){
    ESP_LOGI(UARTAG, "Going backwards");
    int reverse[] = {0x01, 0x05, 0x00, 0x05, 0xFF, 0x00, 0x9C, 0x3B};
    echo_send(uart_num, reverse, sizeof(reverse));
}

void reverseStop(){
    ESP_LOGI(UARTAG, "Stopping");
    int reverse_stop[] = {0x01, 0x05, 0x00, 0x05, 0x00, 0x00, 0xDD, 0xCB};
    echo_send(uart_num, reverse_stop, sizeof(reverse_stop));
}

void setPumpSpeed(int decSpeed){

    char speed[11] = {0x01, 0x10, 0x00, 0x08, 0x00, 0x01, 0x02, 0x00, 0xFA, 0x27, 0x5B};
    speed[8] = decSpeed;
    ESP_LOGI(UARTAG, "Hex speed: %X", decSpeed);
    ESP_LOGI(UARTAG, "size of speed var: %d\n", sizeof(speed));
    echo_send(uart_num, speed, sizeof(speed));
}
```

4.2.5. Raspi LMIC και caching σύστημα

Το λειτουργικό που τρέχει πάνω στο Raspberry Pi 3 είναι το raspbian os, που είναι derivative του Debian linux distro. Για τον χειρισμό του dragino pi hat και την επικοινωνία lorawan χρησιμοποιούμε την βιβλιοθήκη Raspi LMIC. Η συγκεκριμένη βιβλιοθήκη είναι fork

της βιβλιοθήκης LoraMAC in C που σχεδιάστηκε από την IBM για χρήση σε μικροελεγκτές με τα LoRaWAN chips της semtech, στη δική μας περίπτωση το raspi lmic αλλάζει το loramac in c για να μπορεί να λειτουργεί με τα Raspberry Pi.

Το κύριο πρόγραμμα της βιβλιοθήκης μας επιτρέπει να παραμετροποιήσουμε πολλά πράγματα για την επικοινωνία LoRaWAN, όπως τα κανάλια που θέλουμε να μεταδίδονται τα μηνύματα, το Spread Factor, το μεσοδιάστημα στο οποίο θα στέλνονται τα μηνύματα και πολλά άλλα. Στην περίπτωση μου έχω θέσει το μεσοδιάστημα της μετάδοσης στα 5 λεπτά. Έχω κάνει κάποιες αλλαγές στο κύριο πρόγραμμα, μια από τις αλλαγές είναι στο function do_send() το οποίο ελέγχει το μήνυμα που θα σταλεί κατά την μετάδοση.

ttn_otaa.c...

```
void do_send(osjob_t* j) {

        char strTime[16];
        getSystemTime(strTime , sizeof(strTime));
        if (LMIC.opmode & OP_TXRXPEND) {
                printf("OP_TXRXPEND, not sending\n");
        } else {
                if(queueEmpty()){
                        printf("The queue has values\n");
                        digitalWrite(RF_LED_PIN, HIGH);
                        uint8_t *tes = testRead();
                        cout << "In send " << endl;
                        cout << tes << endl;
                        char buf[fileSize * 2 + 1]; /* one extra for \0 */
                        if (to_hex(buf, sizeof(buf), tes, fileSize)){
                                printf("%d", sizeof(tes)*8+2);
                        }
                        LMIC_setTxData2(1, tes, fileSize, 0);
                        printf("Packet queued\n");
                } else {
                        printf("The queue is empty\n");
                        LMIC_setTxData2(1, 00, 1, 0);
                }
        }
}
```

Το function queueEmpty() πολύ απλά τσεκάρει αν το json αρχείο είναι άδειο.



Το function ελέγχει αν το queue μηνυμάτων είναι άδειο και στη περίπτωση αυτή θα κάνει μετάδοση στο lorawan ένα byte 00. Αν υπάρχει μήνυμα στο queue μας τότε καλεί ένα function το οποίο διαβάζει το cache που έχουμε σχεδιάσει και γραμμή προς γραμμή παίρνει τα μηνύματα και τα στέλνει μέσω lorawan στο the things network. Λόγω της ιδιαιτερότητας του LoRaWAN, η μετάδοση που έχουμε θέσει να γίνεται κάθε 5 λεπτά μπορεί να καθυστερήσει και να χρειαστεί παραπάνω από 5 λεπτά, γι αυτό το λόγο έχω σχεδιάσει ένα πολύ απλό συστημα FIFO. Το FIFO αυτό κρατάει σε ένα json αρχείο τα μηνύματα που έρχονται μέσω mqtt απο το wifi κόμβο αισθητήρα και έτσι μπορούν να παραμείνουν στη μνήμη σε περίπτωση που υπάρξει καθυστέρηση για τη μετάδοση. Μαζί με την LMIC βιβλιοθήκη τρέχει και ένα δικό μου πρόγραμμα γραμμένο σε C++ το οποίο χειρίζεται το caching, το mqtt client που επικοινωνεί με τους wifi κόμβους χρησιμοποιώντας την βιβλιοθήκη mosquitto και επίσης κάνει την σειριοποίηση των δεδομένων με protocol buffers για την μετάδοση των δεδομένων.

Στο αρχείο με όνομα mqtt_com.cpp λοιπόν υπάρχουν διάφορα functions που καλυπτουν τις λειτουργίες που ανέφερα. Για αρχή με χρήση της βιβλιοθήκης mosquitto επικοινωνεί με τους wifi κόμβους. Το function on_message τσεκάρει και βλέπει αν το mqtt μήνυμα που ήρθε είναι μήνυμα από τον κόμβο με τον αισθητήρα και αν αυτός ο έλεγχος περάσει τότε καλεί το function save_message() που αποθηκεύει το mqtt μήνυμα στο FIFO που έχουμε φτιάξει για caching.

mqtt_com.c...

```c
void on_message(struct mosquitto *mosq, void *obj, const struct mosquitto_message *msg)
{
        cJSON *json = cJSON_Parse((char *)msg->payload); // load and parse the json message
        if (strcmp(json->child->string, "role") == 0) // Check if the message is from a node and from which node
        {
                if (strcmp(json->child->valuestring, "sensor") == 0)
                {
                        save_message((char *)msg->payload);
                        printf("Soil Sensor\n");
                }
        }
}
```

mqtt_com.c...

```c
void save_message(char *payload)
{
        msgs = fopen(filename, "a");     // Open file for append
        fprintf(msgs, "%s\n", payload); // Append every new mqtt message to message text file
        //fputs(payload, msgs);
        fclose(msgs);
}
```

Το function είναι πολύ απλό καθώς κάνει append κάθε νέο μήνυμα στο τέλος του json αρχείου. Η μεταβλητή filename είναι ίση με “messages.json” και το json αρχείο βρίσκεται στο ίδιο directory με το πρόγραμμα.

4.2.6. Protocol Buffer serialisation/deserialisation, uplink and downlink

Η σειριοποίηση των δεδομένων με τα protocol buffers ξεκινάει από το function readAndDelete(). Το function αυτό ανοίγει το json αρχείο που περιέχει τα cached μηνύματα, δημιουργεί ένα προσωρινό αρχείο και αντιγράφει εκεί το υπάρχον αρχείο χωρίς την πρώτη γραμμή για να αφαιρεθεί από το FIFO list. Η πρώτη γραμμή μπαίνει σε έναν buffer και το προσωρινό αρχείο παίρνει την θέση του κύριου αρχείου. Ο buffer επιστρέφει πίσω στο σημείο που έγινε η κλήση του function ως ένα json object.

mqtt_com.c...

```
cJSON* readAndDelete()
{
        FILE *file, *temp;
        char temp_filename[128];
        char buffer[1024];
        int delete_line = 1;

        strcpy(temp_filename, "temp____");
        strcat(temp_filename, filename);

        file = fopen(filename, "r");
        temp = fopen(temp_filename, "w");

        if (file == NULL || temp == NULL)
        {
                printf("Error opening file(s)\n");
                exit;
        }
        bool keep_reading = true;
        int current_line = 1;
        char testBuffer[500];
        do
          {
                fgets(buffer, 1024, file);
                if (feof(file)){
                        keep_reading = false;
                }
                else if (current_line ≠ delete_line) {

                        fputs(buffer, temp);
                } else if (current_line == delete_line){
                        strcpy(testBuffer, buffer);
                        printf("%s", testBuffer);
                }
                current_line++;
        } while (keep_reading);

        fclose(file);
        fclose(temp);

        remove(filename);
        rename(temp_filename, filename);
        return cJSON_Parse(testBuffer);
}
```

mqtt_com.c...

```
uint8_t *testRead()
{
        cJSON* object = readAndDelete();
        lora::Packet packet;

        cJSON *id = cJSON_GetObjectItemCaseSensitive(object,"id");
        cJSON *msgEpoch= cJSON_GetObjectItemCaseSensitive(object,"epoch");
        if (cJSON_IsString(id) && (id->valuestring != NULL)) {
                packet.set_deviceid(id->valuestring);
        }

        packet.set_cmd(1);
        int epochTime;
        packet.set_epoch((unsigned)time(NULL));
        if (RTC_EPOCH_READ != msgEpoch->valueint)
        {
                packet.set_valuehastimefiff(true);
                packet.set_valueminutesdiff(ceil((msgEpoch->valueint - RTC_EPOCH_READ) / 60));
        }
        else
        {
                packet.set_valuehastimefiff(false);
        }

        cJSON *msgData= cJSON_GetObjectItemCaseSensitive(object,"data");
        cJSON *objTemp= cJSON_GetObjectItemCaseSensitive(msgData,"temp");
        cJSON *objHumi= cJSON_GetObjectItemCaseSensitive(msgData,"vwc");
        cJSON *objEC= cJSON_GetObjectItemCaseSensitive(msgData,"ec");
        packet.set_soiltemp(objTemp->valuedouble);
        packet.set_soilmoisture(objHumi->valuedouble);
        packet.set_eleccontact(objEC->valuedouble);

        fileSize = packet.ByteSizeLong();
        if (fileSize > 50)
        {
                printf("packet is too big");
        }

        cJSON_Delete(id);
        uint8_t *buffer = (uint8_t *)malloc(sizeof(uint8_t) * fileSize);
        packet.SerializeToArray(buffer, fileSize);
        std::cout << "\nBytes: " << buffer;
        free(object);
        return buffer;

}
```

Το function testRead() καλεί το function readAndDelete() και δέχεται το object από την πρώτη θέση του FIFO list, γίνεται αρχικοποίηση του protocol buffer με το protobuf message που έχουμε σχεδιάσει και περιέχει όλες τις μεταβλητές για τις μετρήσεις του αισθητήρα. Ελέγχουμε αν το unix timestamp της μέτρησης έχει διαφορά με το τωρινό unix timestamp του raspberry pi, αν υπάρχει διαφορά τότε η μεταβλητή boolean για διαφορά χρόνου γίνεται true και καταχωρούμε την διαφορά του χρόνου της μέτρησης από τον χρόνο του raspberry pi, έπειτα καταχωρούμε τις μετρήσεις στις μεταβλητές του protobuf message. Στη συνέχεια γίνεται η σειριοποίηση του message και επιστρέφει στο σημείο που έγινε η κλήση του function και τέλος γίνεται το uplink του μηνύματος μέσω του lorawan.

mqtt_com.c...

```
void downSender(char *downlink, int size){
        lora::PumpControl pumpControl;
        if (!pumpControl.ParseFromArray(downlink, size)){
                std::cerr << "Failed to parse message." << std::endl;
        }
        cJSON *pumpControlJson = cJSON_CreateObject();
        if (pumpControlJson == NULL) {
                printf("Failed to create JSON object.\n");
        }
        cJSON_AddNumberToObject(pumpControlJson, "cmd", pumpControl.cmd());
        cJSON_AddStringToObject(pumpControlJson, "deviceId", pumpControl.deviceid().c_str());
        cJSON_AddNumberToObject(pumpControlJson, "speed", pumpControl.speed());
        cJSON_AddNumberToObject(pumpControlJson, "duration", pumpControl.duration());
        cJSON_AddNumberToObject(pumpControlJson, "interval", pumpControl.interval());
        // Print the JSON object as a string
        char *jsonString = cJSON_Print(pumpControlJson);
        if (jsonString == NULL) {
                printf("Failed to print JSON string.\n");
                cJSON_Delete(pumpControlJson);
        }
        mosquitto_publish(mosq, NULL, "/GH_1/pump", strlen(jsonString), jsonString, 0, false);
        free(jsonString);
}
```

Στην περίπτωση που το lmic δεχτεί downlink από το lorawan το μήνυμα δίνεται στο function downSender ως παράμετρος. Το μήνυμα από το downlink χρειάζεται το δικό του protobuf message για να γίνει η αποσειριοποίηση οπότε κάνουμε την αρχικοποίηση του message pumpControl. Δημιουργούμε ένα json object και χρησιμοποιώντας το compiled αρχείο του protobuf αποσειριοποιούμε τα δεδομένα και τα καταχωρούμε στο json object. Τέλος το json object στέλνεται στον wifi κόμβο με την αντλία χρησιμοποιώντας το publish function του mosquitto library.

4.2.7. Backend NodeJS Server

Για τον backend server επέλεξα να χρησιμοποιήσω την πλατφόρμα nodejs και να γράψω τον server με javascript λόγω της ευκολίας που προσφέρει. Χρησιμοποίησα τις npm βιβλιοθήκες mqtt, protobufjs, και mongodb. Για αρχή συνδέθηκα στον application server του The Things Network μέσω mqtt σε 2 συγκεκριμένα topics, το ένα για να μπορεί το backend να δέχεται τα uplinks που στέλνονται από το lorawan end device και το άλλο topic για να μπορεί ο backend server να κάνει downlink για την αλλαγή των παραμέτρων της αντλίας για το πότισμα.

MQTT Topics

backend....

```
const soilSensorUplinkTopic = 'v3/bb-data-reader@tei-of-crete/devices/wisnode-1/up'
const downlink = 'v3/bb-data-reader@tei-of-crete/devices/wisnode-1/down/push'

const client = mqtt.connect(connectUrl, {
  clientId,
  clean: true,
  connectTimeout: 4000,
  username: "bb-data-reader@tei-of-crete",
  password:
    "****",
  reconnectPeriod: 1000
});
const downlinkClient = mqtt.connect(connectUrl, {
  clientId,
  clean: true,
  connectTimeout: 4000,
  username: "bb-data-reader@tei-of-crete",
  password:
    "****",
  reconnectPeriod: 1000
});
```

Ο κώδικας χρησιμοποιεί την mqtt βιβλιοθήκη και συνδέεται στα topics για τα uplinks και τα downlinks.

Χειρισμός uplink

backend....

```
    client.subscribe([soilSensorUplinkTopic], () => {
        console.log(`Subscribe to topic '${soilSensorUplinkTopic}'`)
        client.on('message', (topic, message) => {
            testMsg = JSON.parse(String(message));
            // Put it in an array for spliting it up
            binaryArray = testMsg.uplink_message.decoded_payload.bytes;
            binaryArray.forEach(element => {
                // Add leading zeros to those who dont have them, just in case.
                let hexNum = element.toString(16).toUpperCase().padStart(2, '0');
                hexArray.push(hexNum); // Push the number in the array
            });
            let newMessage = hexArray.join('');
            hexArray = [];
```

Χρησιμοποιούμε το mqtt client που είναι subscribed στο topic των uplink για να διαβάσουμε το μήνυμα που έρχεται από το uplink του wifi κόμβου με τον αισθητήρα. Το uplink μας στέλνει ενα json object και θα πρέπει να το κάνουμε parse και να κρατήσουμε το σωστό field που περιέχει το πραγματικό μήνυμα από το end device μας, το field που μας ενδιαφέρει είναι το uplink_message.decoded_payload.bytes το οποίο είναι ένας πίνακας από hex bytes που θα πρέπει να καθαρίσουμε και να μετατρέψουμε σε χρήση μορφή.

backend....

```
if (newMessage == 0 || newMessage == 8) {
    log("Message is empty, do nothing");
} else {
    try {
        // Use the hex option in order to not fuck up the hex string
        const decodedMessage = Packet.decode(Buffer.from(newMessage, 'hex'));
        console.log(decodedMessage);
        saveMessageToDB(decodedMessage);
        noderedClient.publish(sensordataToNodered, JSON.stringify(decodedMessage));

        if ( decodedMessage.soilMoisture > threshold3.moist) {
            log("thres 3");
            sendDownlink(threshold3.speed, threshold3.time, threshold3.inter);
        } else if (decodedMessage.soilMoisture > threshold2.moist) {
            log("thres 2");
            sendDownlink(threshold2.speed, threshold2.time, threshold2.inter);
        } else if (decodedMessage.soilMoisture > threshold1.moist) {
            log("thres 1");
            sendDownlink(threshold1.speed, threshold1.time, threshold1.inter);
        } else {
        }
    } catch (e) {
        if (e instanceof protobuf.util.ProtocolError) {
            // e.instance holds the so far decoded message with missing required fields
            console.log("Error" + e);
        } else {
            // wire format is invalid
            console.log("Wire format is invalid", e);
        }
    }
}
```

Στην συνέχεια κάνουμε έναν έλεγχο για να δούμε αν το μήνυμα που ήρθε είναι το byte 00 ή 08 που σημαίνει ότι το μήνυμα μας είναι άδειο. Αν το μήνυμα δεν είναι άδειο τότε προσπαθούμε να κάνουμε αποσειριοποίηση το hex array χρησιμοποιώντας το .proto αρχείο που έχουμε σχεδιάσει. Αφού γίνει η αποσειριοποίηση στέλνουμε το μήνυμα στο function saveMessageToDB() και αποθηκεύεται σε μια βάση δεδομένων MongoDB έτσι ώστε να μπορεί να γίνει αργότερα αξιολόγηση των δεδομένων και οποιαδήποτε περαιτέρω επεξεργασία. Έπειτα, γίνεται η χρήση κάποιων ρυθμίσεων threshold που έχουν οριστεί από τον χρήστη του συστήματος μέσω της διεπαφής χρήστη που σχεδιάστηκε με node red. Ο

backend server συνδέεται με mqtt με τη διεπαφή χρήστη. Ο backend server δέχεται μηνύματα json μέσω της mqtt επικοινωνίας και έτσι θέτει τις παραμέτρους κάθε threshold.

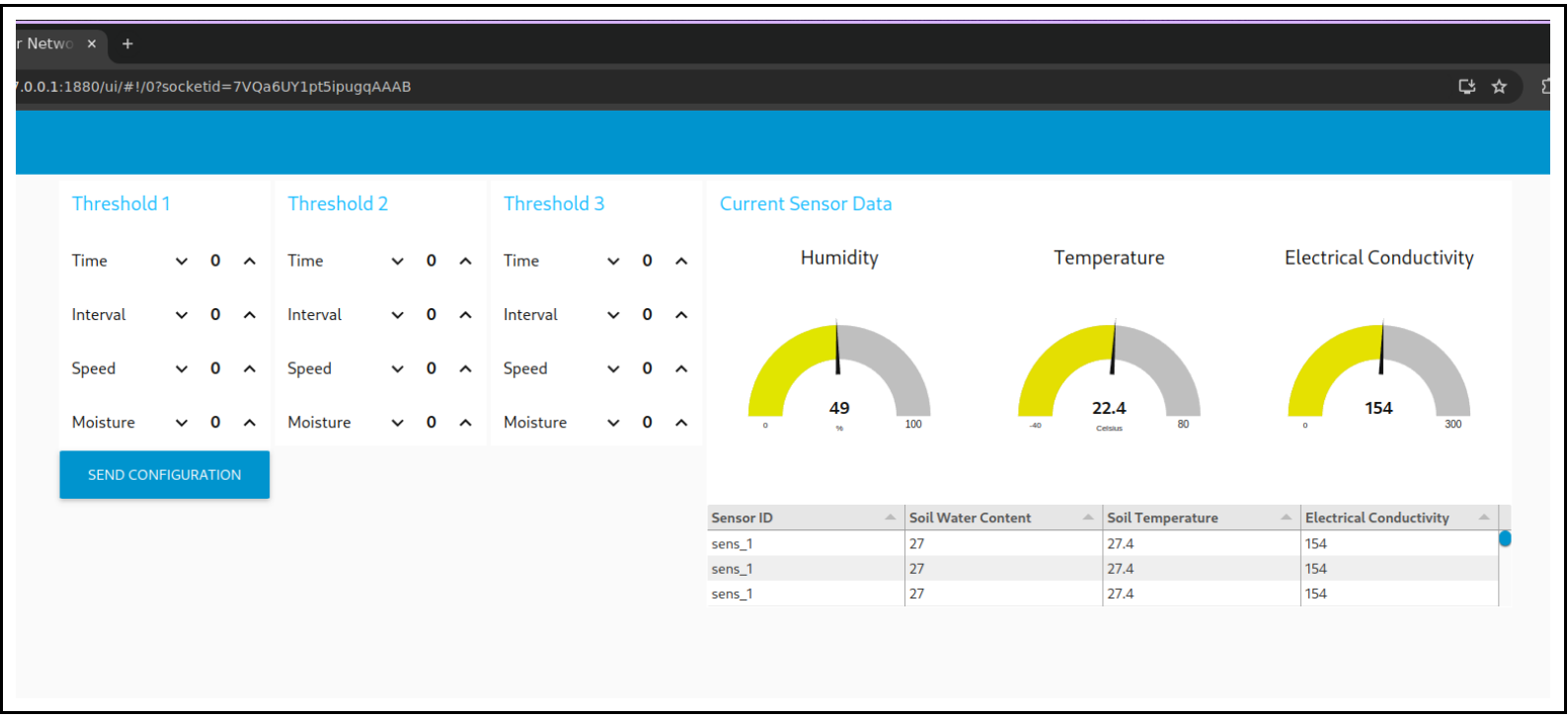
backend....

```
noderedClient.on('connect', () => {
    console.log("nodered broker connected");
    noderedClient.subscribe([configThresholdTopic], () => {
        console.log(`Subscribe to topic '${configThresholdTopic}'`)
        noderedClient.on('message', (topic, message) => {
            log(JSON.parse(message));
            let newMessage = JSON.parse(message);
            threshold1.moist = newMessage.thres1[0];
            threshold1.speed = newMessage.thres1[1];
            threshold1.time = newMessage.thres1[2];
            threshold1.inter = newMessage.thres1[3];

            threshold2.moist = newMessage.thres2[0];
            threshold2.speed = newMessage.thres2[1];
            threshold2.time = newMessage.thres2[2];
            threshold2.inter = newMessage.thres2[3];

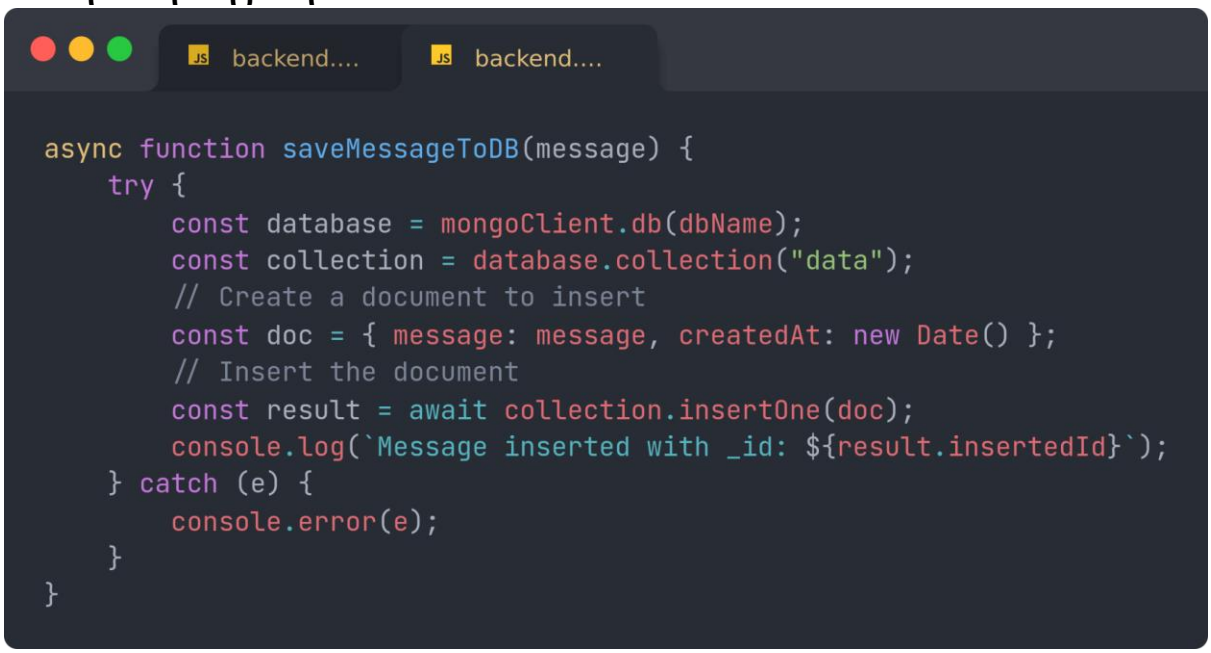
            threshold3.moist = newMessage.thres3[0];
            threshold3.speed = newMessage.thres3[1];
            threshold3.time = newMessage.thres3[2];
            threshold3.inter = newMessage.thres3[3];
        })
    });
})
```

Διεπαφή Χρήστη με Node Red



Στην διεπαφή, ο χρήστης έχει την δυνατότητα να ελέγξει και να αλλάξει τις παραμέτρους κάθε threshold για το πότισμα αλλάζοντας τις αριθμητικές τιμές. Επίσης του παρέχονται εργαλεία για την παρακολούθηση των τιμών που έχουν διαβάσει οι αισθητήρες, γίνεται εμφάνιση των δεδομένων που διαβάζονται εκείνη τη στιγμή σε μετρητές και παλαιότερες τιμές σε έναν πίνακα τιμών. Γίνεται η χρήση 3 threshold ή αλλιώς περιοχών τιμής υγρασίας. Τα thresholds θέτονται από τον χρήστη, για παράδειγμα το πρώτο και πιο χαμηλό threshold μπορεί να έχει τιμή υγρασίας 25%, το δεύτερο 50% και το τελευταίο 75%. Για κάθε ένα από αυτές τις τιμές ο χρήστης καθορίζει τον χρόνο ποτίσματος το μεσοδιάστημα κατά το οποίο η αντλία ποτίσματος θα βρίσκεται σε αδράνεια και την ταχύτητα με την οποία θα περιστρέφεται η αντλία όταν ποτίζει. Για παράδειγμα αν ο χρήστης θέλει να κρατήσει την υγρασία του εδάφους στο 50% θα πρέπει να θέσει μεγάλες τιμές ποτίσματος στο πρώτο threshold και χαμηλές τιμές ποτίσματος στο τρίτο threshold έτσι ώστε να περιοριστεί στο δεύτερο.

Αποθήκευση στη βάση



Το function για την αποθήκευση στη βάση δεδομένων είναι αρκετά απλό, εφόσον έχουμε συνδεθεί στην βάση με το username, το password και το όνομα της βάσης, διευκρινίζουμε το όνομα του collection στο οποίο θα μείνουν τα δεδομένα. Δημιουργούμε ένα νέο json object με τα δεδομένα που θέλουμε να σώσουμε και τη χρονική στιγμή που δημιουργήθηκε το object και τέλος εισάγουμε το json object στη βάση.

Αποστολή downlink

backend.... backend....

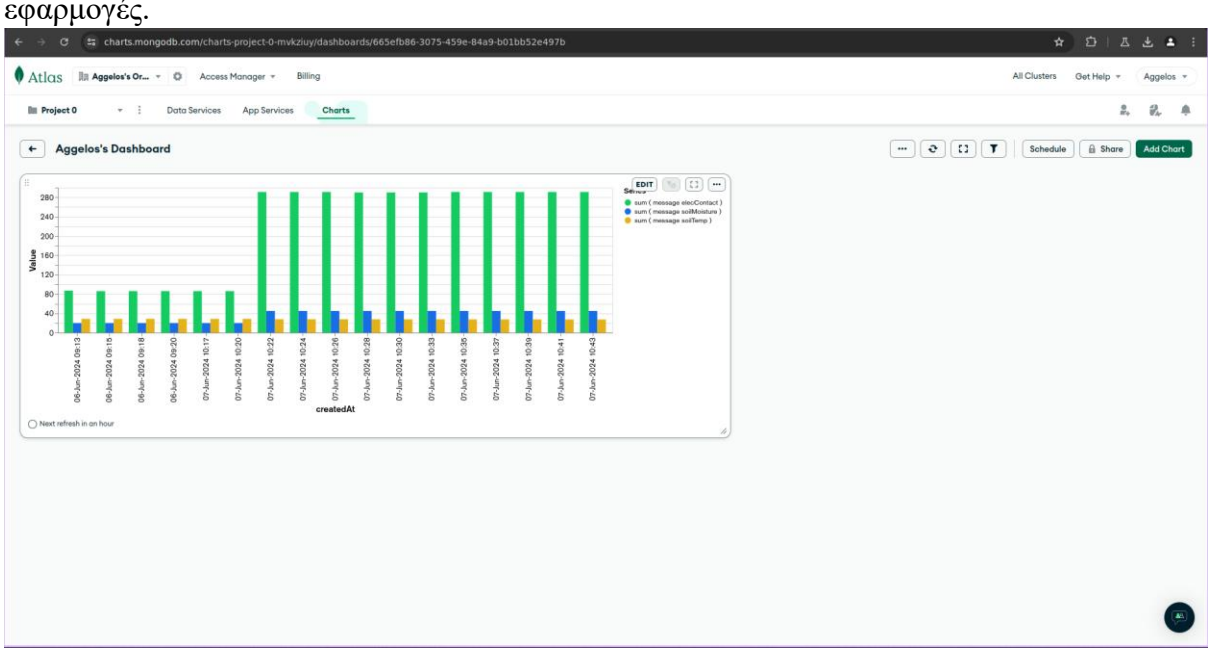
```
function sendDownlink(speed, interval, duration) {
    console.log("we're in downlink now");
    console.log(btoa("test"));
    let buffer;
    let mesg;
    let downlinkMsg = {
        cmd : 1,
        deviceId : "pump_1",
        speed : speed,
        duration : duration,
        interval : interval
    };
    try {
        buffer = DownlinkPacket.encode(downlinkMsg).finish();
        mesg = Buffer.from(buffer, 'hex').toString('base64');
    } catch (e) {
        if (e instanceof protobuf.util.ProtocolError) {
            console.log("Error" + e);
        } else {
            // wire format is invalid
            console.log("Wire format is invalid", e);
        }
    }
    let newSettings = 1;
    if(newSettings){
        // In order to send a downlink we have to send this specific line of text.
        // frm_payload is going to be our message which has to be in base64,
        // we transform out text to base64 using the btoa function
        downlinkClient.publish(downlink,
        '{"downlinks":[{"f_port": 15,"frm_payload":"'+ mesg +'","priority": "NORMAL"}]}')
    }
}
```

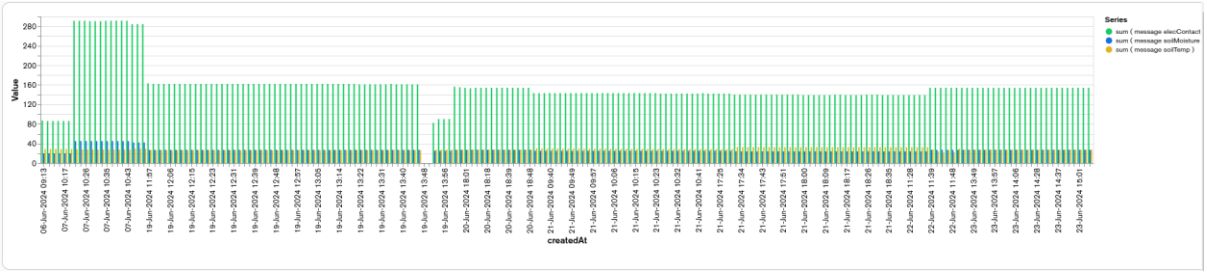
Το function για την δημιουργία και αποστολή του downlink δέχεται 3 παραμέτρους, τη ταχύτητα, το διάστημα για το οποίο θα κινείται η αντλία και το μεσοδιάστημα για το οποίο η αντλία θα είναι σε αδράνεια. Για να σειριοποιήσουμε το μήνυμα μας πρέπει να δημιουργήσουμε ένα json object και να το γεμίσουμε με τα δεδομένα που θέλουμε να στείλουμε, έπειτα χρησιμοποιούμε πάλι το .proto αρχείο που έχουμε σχεδιάσει αλλά τώρα

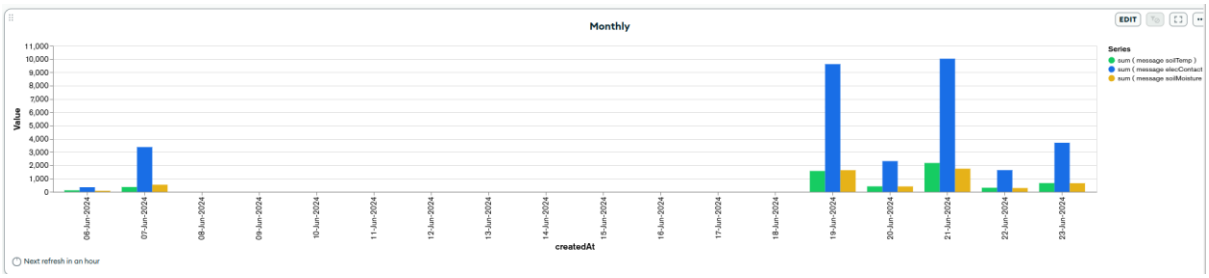
χρησιμοποιούμε το protobuf message που σχεδιάσαμε για τον κόμβο με την αντλία. Κάνουμε την σειριοποίηση με το function Downlink.encode(), μετατρέπουμε το output της σειριοποίησης σε base64 και βάζουμε αυτό το αποτέλεσμα σε ένα ακόμα json object για να σταλεί στο downlink queue της The Things Network με τη χρήση mqtt publish στο topic του downlink.

Data visualisation

Για την παρακολούθηση των δεδομένων μπορούμε να χρησιμοποιήσουμε τα charts του MongoDB Atlas που το προσφέρει η βάση δεδομένων. Το MongoDB Atlas είναι μια υπηρεσία βάσης δεδομένων στο cloud που παρέχεται από την MongoDB, Inc. Η υπηρεσία παρέχει διάφορες βασικές λειτουργίες όπως την διαχείριση δεδομένων, δημιουργία αντιγράφων ασφαλείας παρακολούθηση των δεδομένων. Ένα πολύ σημαντικό χαρακτηριστικό είναι η δημιουργία γραφημάτων χωρίς την χρήση κώδικα το οποίο δίνει την δυνατότητα δημιουργίας γραφημάτων στον χρήστη χωρίς την βοήθεια του προγραμματιστή. Οι πίνακες είναι διαδραστικοί, ενημερώνονται σε πραγματικό χρόνο και μπορούν να ενσωματωθούν σε άλλες εφαρμογές.







5. Συμπεράσματα και μελλοντική εργασία

Σε αυτή την εργασία, εφαρμόσαμε επιτυχώς ένα σύστημα επικοινωνίας και ελέγχου για την παρακολούθηση φυτών, χρησιμοποιώντας τα Protocol Buffers για την σειριοποίηση των δεδομένων κατά την μετάδοση τους μέσω LoRaWAN για την αποστολή τους σε μεγάλες αποστάσεις, ενσωματώνοντας αισθητήρες και ενεργοποιητές σειριακής επικοινωνίας RS-485. Η χρήση του Raspberry Pi 3 σε συνδυασμό με το Dragino Pi HAT αποδείχτηκε να είναι αρκετά καλή λύση για την δημιουργία ενός LoRaWAN end device για την αξιόπιστη μετάδοση δεδομένων. Η χρήση των Protocol Buffers για την σειριοποίηση των δεδομένων εξασφάλισε αποτελεσματική σειριοποίηση, μειώνοντας το μέγεθος των μηνυμάτων σε σχέση με άλλες τεχνικές και βελτιώνοντας τη μετάδοση τους. Το σύστημα συνδυάζοντας όλες αυτές τις τεχνολογίες και τεχνικές αναδεικνύει την δυνατότητα χρήσης του για βιομηχανικές εφαρμογές όπου η μετάδοση δεδομένων σε μεγάλες αποστάσεις και ο αποτελεσματικός χειρισμός των δεδομένων είναι κρίσιμα γνωρίσματα.

Η μελλοντική δουλειά θα μπορούσε να συγκεντρωθεί σε αρκετούς τομείς οι οποίοι θα αυξήσουν τις δυνατότητες του συστήματος αλλά και θα μειώσουν αρκετούς από τους περιορισμούς του. Ένας τομέας που θα καλυτέρευε το σύστημα είναι η αλλαγή του δημόσιου LoRaWAN δικτύου σε ένα ιδιωτικό δίκτυο LoRaWAN έτσι ώστε να υπάρχει πλήρης έλεγχος των συσκευών που εκπέμπουν και των καναλιών στα οποία γίνεται η εκπομπή. Έτσι θα μπορούν να μεταδίδονται παραπάνω μηνύματα και σε μεγαλύτερο ρυθμό και χωρίς τον φόβο για παρεμβολή ή σύγκρουση μηνυμάτων με άλλους χρήστες. Κάτι ακόμα που θα έκανε το σύστημα καλύτερο θα ήταν η έρευνα για την παροχή ενέργειας των WiFi κόμβων, για παράδειγμα θα ήταν εξαιρετική ιδέα η χρήση μπαταριών και μικρών φωτοβολταϊκών για την τροφοδοσία τους για μεγάλα χρονικά διαστήματα χωρίς την ανάγκη κεντρικής παροχής ενέργειας.

Αναφορές

[1]. Protocol Buffers, "Overview," Protocol Buffers Documentation, May 18, 2024. [Online]. Available: https://protobuf.dev/overview/. [Accessed: May 18, 2024].

[2]. Apache Software Foundation, "Apache Thrift - Home," Apache Thrift. [Online]. Available: https://thrift.apache.org/. [Accessed: May 18, 2024].

[3]. Apache Software Foundation, "Apache Avro 1.11.1 Documentation," Apache Avro. [Online]. Available: https://avro.apache.org/docs/1.11.1/. [Accessed: May 18, 2024].

[4]. J. Hopmans, "Use of dielectric properties to measure soil water content," University of California, 2018. [Online]. Available: https://ucmanagedrought.ucdavis.edu/PDF/DROUGHT_WEB%20_DIELECTRIC.pdf.

[5]. Geopard, "What Are the Types of Sensors Used in Agriculture?," Geopard. [Online]. Available: https://geopard.tech/blog/what-are-the-types-of-sensors-used-in-agriculture/. [Accessed: May 20, 2024].

[6]. Prem Rajak a, Abhratanu Ganguly a , Satadal Adhikary b , Suchandra Bhattacharya c, "Internet of Things and smart sensors in agriculture: Scopes and challenges", Journal of Agriculture and Food Research, Available: URL. [Accessed: May 20, 2024].

[7]. Wikipedia contributors, "Wireless sensor network," *Wikipedia, The Free Encyclopedia*, 21-May-2024. [Online]. Available: https://en.wikipedia.org/wiki/Wireless_sensor_network. [Accessed: 21-May-2024].

[8]. Wikipedia contributors, "Internet of things," *Wikipedia, The Free Encyclopedia*, 21-May-2024. [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things. [Accessed: 21-May-2024].

[9]. Wikipedia contributors, "Ubiquitous computing," *Wikipedia, The Free Encyclopedia*, 21-May-2024. [Online]. Available: https://en.wikipedia.org/wiki/Ubiquitous_computing. [Accessed: 21-May-2024].

[10]. Wikipedia contributors, "Amazon Echo," *Wikipedia, The Free Encyclopedia*, 21-May-2024. [Online]. Available:https://en.wikipedia.org/wiki/Amazon_Echo. [Accessed: 21-May-2024].

[11]. Wikipedia contributors, "Google Nest," *Wikipedia, The Free Encyclopedia*, 21-May-2024. [Online]. Available: https://en.wikipedia.org/wiki/Google_Nest_(smart_speakers). [Accessed: 21-May-2024].

[12]. Wikipedia, "Machine to machine," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Machine_to_machine. Accessed: May 22, 2024.

[13]. J. Lee, B. Bagheri and H. A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18-23, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527320301559. Accessed: May 22, 2024.

[14]. "Multidrop bus," Wikipedia. https://en.wikipedia.org/wiki/Multidrop_bus (accessed May 22, 2024).

[15] "How Far And How Fast Can You Go With RS-485", Maxim Integrated. https://pdfserv.maximintegrated.com/en/an/AN3884.pdf (accessed May 22, 2024).

[16] "Spread spectrum," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Spread_spectrum. Accessed: 2024, May 27.

[17] "LoRa Modulation Basics" (PDF). *Semtech*. Archived from the original (PDF) on 2019-07-18.
Available:[https://web.archive.org/web/20190718200516/https://www.semtech.com/uploads/documents/an1200.22.pdf].Accessed: 2024, May 27.

[18] "SX1261/2 Datasheet". *Semtech SX1276*. Available: [https://www.semtech.com/products/wireless-rf/lora-connect/sx1262#download-resources].Semtech. Accessed: 2024, May 27.

[19]M. Chiani and A. Elzanaty, "On the LoRa Modulation for IoT: Waveform Properties and Spectral Analysis," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8463-8470, Oct. 2019, doi: 10.1109/JIOT.2019.2919151.

keywords: {Frequency modulation;Internet of Things;Time-domain analysis;Time-frequency analysis;Chirp modulation;Digital modulation;Internet of Things (IoT);low power long range (LoRa) modulation;power spectral density} Available at: [https://ieeexplore.ieee.org/document/8723130]. Accessed: 2024, May 27.

[20] "Link Budget," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Link_budget. Accessed: 2024, May 27.

[21]. J.C. Liando; A. Gamage; A.W. Tengourtius; M. Li (2019). "Known and Unknown Facts of LoRa: Experiences from a Large-Scale Measurement Study". *ACM Transactions on Sensor Networks*. 15 (2): Article No. 16, pp 1–35. doi:10.1145/3293534. hdl:10356/142869. ISSN 1550-4859. S2CID 53669421. Accessed: 2024, May 27

[22]. "What are LoRa® and LoRaWAN®?". *LoRa Developer Portal*. Available at: [https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf]. Accessed: 2024, May 27.

[23]. Priva, "Priva: Leading in Smart Building Automation and Horticulture Solutions," [Online]. Available: https://www.priva.com/. Accessed: 03 June 2024

[24]. Aranet, "Aranet: Wireless Environmental Monitoring Solutions," [Online]. Available: https://www.aranet.com/. Accessed: 03 June 2024.

[25]. "gRPC FAQ," gRPC. [Online]. Available: https://grpc.io/docs/what-is-grpc/faq/. [Accessed: 12-June-2024].

[26]. Elecrow, "MAX485 Datasheet," accessed June 14, 2024. [Online]. Available: https://www.elecrow.com/download/product/CIA00485T/MAX485.pdf