ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

MSc in ROBOTICS

Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών

Πρόγραμμα Μεταπτυχιακών Σπουδών στη Ρομποτική

Διπλωματική Εργασία

Υλοποίηση Εφαρμογής IoT για την βέλτιστη χρήση πόρων σε έξυπνες πόλεις

Implementation of IoT application for optimum resource usage in smart cities

Μεταπτυχιακός Φοιτητής: Ιωάννης Χρυσοχοΐδης
ΑΜ: 106
Επιβλέπων Καθηγητής: Ιωάννης Βουρβουλάκης

Σέρρες, Ιούνιος 2023

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή κ. Ιωάννη Βουρβουλάκη, που μου έδωσε την δυνατότητα να ασχοληθώ με ένα τόσο επίκαιρο θέμα και με καθοδήγησε – υποστήριξε καθ' όλη τη διάρκεια της διπλωματικής εργασίας. Με τη πολύτιμη συνεισφορά αλλά και την επίβλεψη του, κατάφερα να αντιμετωπίσω και να ξεπεράσω όλες τις προκλήσεις που συνάντησα κατά την υλοποίηση.

Επιπλέον, το μάθημα «Ενσωματωμένα συστήματα – P101» αποτέλεσε ένα πολύ σημαντικό σημείο αναφοράς, έτσι ώστε να μπορέσω να αποκτήσω τις απαιτούμενες γνώσεις, οι οποίες ήταν απαραίτητες για τη διεκπεραίωση της διπλωματικής μου εργασίας.

Στην συνέχεια, θα ήθελα να ευχαριστήσω την οικογένεια μου, για την αμέριστη στήριξη της όλα αυτά τα χρόνια, τόσο κατά τη διάρκεια των προπτυχιακών, όσο και κατά την διάρκεια των μεταπτυχιακών μου σπουδών.

Τέλος, θα ήθελα να ευχαριστήσω την υπόλοιπη τριμελής επιτροπή τους κ.κ. Βολογιαννίδη Σταύρο και Νικολαΐδη Αθανάσιο, για τον χρόνο που διέθεσαν κατά την παρουσίαση της διπλωματικής εργασίας μου όσο και για τις γνώσεις που μου μετέδωσαν κατά την διάρκεια των μεταπτυχιακών μου σπουδών.

Ιωάννης Χρυσοχοΐδης

Περιεχόμενα

Περίληψη

Το διαδίκτυο των αντικειμένων (IoT) αποτελεί αναμφίβολα μια τεχνολογία που αυξάνεται συνεχώς με εκθετικό ρυθμό. Σε αυτό έχουν συμβάλει, η ανάπτυξη των δικτύων πέμπτης γενιάς 5G της κινητής τηλεφωνίας, καθώς και η μεγάλη ποικιλία αισθητήρων συλλογής και αποστολής δεδομένων με χαμηλό κόστος.

Η ανάπτυξη, η χρήση και η σύνδεση στο διαδίκτυο πληθώρας αισθητήρων, οι οποίοι έχουν τη δυνατότητα να συλλέγουν και να αποστέλλουν δεδομένα σε πραγματικό χρόνο, έχει συμβάλει καθοριστικά στην ανάπτυξη του Internet of Things.

Για να χαρακτηριστεί μία πόλη έξυπνη (Smart City), θα πρέπει να μπορεί να διαχειριστεί και να αξιοποιεί την τεράστια ποσότητα δεδομένων που συλλέγονται από τους αισθητήρες, έτσι ώστε να λαμβάνονται γρήγορα αποφάσεις με στόχο την επίτευξη εξοικονόμησης ενέργειας, αλλά και τη βελτιωθεί της ποιότητας ζωής των κατοίκων – επισκεπτών της.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μίας εφαρμογής η οποία θα διαβάζει μετρήσεις ενέργειας και μετρήσεις συνθηκών περιβάλλοντος από κατάλληλους αισθητήρες. Στη συνέχεια, θα τις αποστέλλει με τη χρήση του RabbitMQ στον ελεγκτή Raspberry PI και έπειτα θα δρομολογούνται με τη βοήθεια του πρωτοκόλλου MQTT, για μόνιμη αποθήκευση και απεικόνιση, με χρήση γραφημάτων, στο λογισμικό Thingsboard. Τέλος, στόχος της εργασίας επίσης αποτελεί, ο σχεδιασμός πλακέτας PCB για το μετρητή περιβάλλοντος, χρησιμοποιώντας το λογισμικό KiCad, ώστε η εφαρμογή να προσεγγίζει τη μορφή βιομηχανικού προϊόντος.

Λέξεις Κλειδιά: Διαδίκτυο των αντικειμένων, IoT, πρωτόκολλα επικοινωνία, HTTP, MQTT, AMQP, Modbus, ενεργειακός αναλυτής, μετρητής περιβάλλοντος, Raspberry Pi. Bosch BME 280, Janitza D21 Enegry Meter, KiCad, PCB, σχεδίαση πλακέτας

Abstract

The Internet of Things (IoT) is definitely a technology that is growing at an exponential rate. The development of fifth-generation 5G mobile networks, as well as the wide variety of low-cost sensors for data collection and transmission, have contributed to this particular development.

The development, use, and connection to the Internet of a variety of sensors, which have the ability to collect and send data in real-time, has been instrumental in the development of the Internet of Things.

In order to be classified as a Smart City, a city must be able to manage and use the huge amount of data collected by sensors, so that decisions can be made quickly to achieve energy savings and improve the quality of life of its residents - visitors.

The purpose of this project is to develop an application that will read energy and environmental measurements from appropriate sensors. Then, the application will send them using RabbitMQ to the Raspberry PI controller. Afterward, they will be routed using the MQTT protocol, for permanent storage and visualization using Thingsboard software.

Keywords: Internet of Things, IoT, communication protocols, HTTP, MQTT, AMQP, Modbus, energy meter, environmental meter, Raspberry Pi, Bosch BME 280, Janitza D21 Enegry Meter, KiCad, PCB design

Λίστα εικόνων

Λίστα πινάκων

Κεφάλαιο 1. Διαδίκτυο των Αντικειμένων (Internet of Things)

1.1. Ορισμός IoT

Το διαδίκτυο των αντικειμένων (IoT) αποτελεί έναν από τους ταχύτερα αναπτυσσόμενους κλάδους, με εφαρμογή σε ποικίλους τομείς. Είναι ένα δίκτυο, με ραγδαία ανάπτυξη διασυνδεδεμένων συσκευών, τα οποία μπορούν να ανταλλάσσουν δεδομένα και να επικοινωνούν μεταξύ τους. Πρόκειται για ένα από τα πιο διαδεδομένα αντικείμενα, που σχετίζεται με πολυάριθμους τομείς και συνεχώς εξελίσσεται. Συνδυάζει ένα ευρύ φάσμα εφαρμογών, συστημάτων και αισθητήρων, όπου καταφέρνει να αξιοποιήσει τις δυνατότητες ανάλυσης και επεξεργασίας μεγάλου όγκου δεδομένων, που απαιτούν ισχυρή υπολογιστική ικανότητα. Σε συνδυασμό με την χρήση μικρότερων σε μέγεθος εξαρτημάτων, είναι σε θέση να προσφέρει σημαντικές λύσεις όπου στο παρελθόν θεωρούνταν ανέφικτες. Αν μελετήσει κάποιος την διεθνή βιβλιογραφία εύκολα μπορεί να διαπιστώσει ότι πρακτικά συνεδρίων, άρθρα και επιστημονικές εκθέσεις έχουν ασχοληθεί και μελετήσει την συγκεκριμένη τεχνολογία καθώς καταφέρνει να συνδυάσει αρκετούς κλάδους μεταξύ τους και να προσφέρει σημαντικές λύσεις [1].

Για πρώτη φορά ο όρος ‘’Διαδίκτυο των αντικειμένων’’ χρησιμοποιήθηκε το 1999 από τον βρετανό Kevin Ashton, πρωτοπόρο σε ζητήματα τεχνολογίας. Ο Ashton χρησιμοποίησε για πρώτη φορά τον όρο "Internet of Things" σε μια παρουσίαση που έκανε στην Procter & Gamble το 1999, όπου ανέλυσε τον τρόπο με τον οποίο η τεχνολογία RFID θα μπορούσε να χρησιμοποιηθεί για τη σύνδεση καθημερινών αντικειμένων στο διαδίκτυο. Οραματίστηκε ένα μέλλον όπου τα αντικείμενα θα μπορούσαν να ταυτοποιούνται και να παρακολουθούνται μοναδικά, επιτρέποντας τη βελτίωση της διαχείρισης των αποθεμάτων, τη βελτιστοποίηση της αλυσίδας εφοδιασμού και τη βελτίωση της αποδοτικότητα των διαδικασιών [2].

1.2. Βασικά Χαρακτηριστικά IoT

Μερικά από τα βασικά χαρακτηριστικά που χαρακτηρίζουν το IoT είναι:

- ***Υπηρεσίες σε αντικείμενα:*** Το IoT προσφέρει υπηρεσίες σε αντικείμενα που μπορεί να είναι είτε φυσικές είτε εικονικές συσκευές. Οι συσκευές IoT μπορούν να προσφέρουν μια σειρά από υπηρεσίες και εφαρμογές σε διάφορους τομείς, όπως ο οικιακός αυτοματισμός, η υγειονομική περίθαλψη, οι μεταφορές και η μεταποίηση. Οι συσκευές IoT συλλέγουν και ανταλλάσσουν δεδομένα μεταξύ τους, καθιστώντας δυνατή την παρακολούθηση, την αυτοματοποίηση και τη βελτιστοποίηση των διαδικασιών, εξοικονομώντας χρόνο και πόρους και βελτιώνοντας την ποιότητα των υπηρεσιών [3].
- ***Ανομοιογένεια συσκευών:*** Στην περίπτωση όπου χρησιμοποιηθούν και συνδεθούν διαφορετικές συσκευές μεταξύ τους, τότε παρατηρείται ανομοιογένεια. Στόχος είναι να αλληλοεπιδρούν και να επικοινωνούν με άλλες συσκευές που είναι συνδεδεμένες στο δίκτυο και είναι διαφορετικού κατασκευαστή και μοντέλου [4].
- ***Επικοινωνία:*** Οι συσκευές καλούνται να επικοινωνούν μεταξύ τους, κάνοντας χρήση κατάλληλων πρωτοκόλλων επικοινωνίας, με σκοπό να μπορούν να διαχειριστούν το μεγάλο όγκο δεδομένων αλλά και να εξάγουν ωφέλιμα συμπεράσματα από τα συγκεκριμένα δεδομένα [5].
- ***Ευελιξία:*** Να διαθέτουν την δυνατότητα τροποποίησης, με τέτοιο τρόπο ώστε να μην επηρεάζουν την γενικότερη λειτουργία τόσο του συστήματος όσο και της σύνδεσης με τις άλλες συσκευές [6].
- ***Ελαχιστοποίηση κόστους:*** Τόσο οι κατασκευαστές του IoT όσο και η ίδια η πολιτεία προσπαθεί να ελαχιστοποιήσει το κόστος παραγωγής-συντήρησης αλλά και μείωση της κατανάλωσης ενέργειας [7].
- ***Ποιότητα υπηρεσιών:*** Το IoT θα πρέπει να διαθέτει κατάλληλη ποιότητα υπηρεσιών δηλαδή να δημιουργούνται εφαρμογές με υψηλή ποιότητα κυρίως αν πρόκειται για υπηρεσίες που πραγματοποιούνται σε πραγματικό χρόνο [8].
- ***Ασφάλεια:*** Αποτελεί ίσως το πιο σημαντικό χαρακτηριστικό του IoT. Η διασφάλιση και η προστασία των προσωπικών δεδομένων των χρηστών είναι ζωτικής σημασίας. Συνεπώς, δίνεται ιδιαίτερη σημασία από την πολιτεία, στο να κατασκευαστούν συστήματα και συσκευές, ώστε να διασφαλίζεται το κομμάτι της ασφάλειας κατά τη σύνδεση στο δίκτυο [9].

1.3. Πλεονεκτήματα Χρήσης IoT

Το Διαδίκτυο των αντικειμένων προσφέρει αρκετά πλεονεκτήματα στην καθημερινή μας ζωή και παρατίθενται παρακάτω.

- **Αυτοματοποίηση εργασιών & εξοικονόμηση χρόνου**: Καθώς οι συσκευές IoT αλληλεπιδρούν και επικοινωνούν μεταξύ τους, μπορούν να αυτοματοποιήσουν τις διεργασίες, βοηθώντας στη βελτίωση της ποιότητας των υπηρεσιών και μειώνοντας την ανάγκη για ανθρώπινη παρέμβαση [10] [11].
- **Απομακρυσμένος έλεγχος συσκευών**: Αποτελεσματικότερος έλεγχος συσκευών, δίχως τον περιορισμό χρόνου και χώρου και πάνω από όλα σε πραγματικό χρόνο 24/7. Αυτό σημαίνει, ότι έχουμε τη δυνατότητα να ελέγχουμε τις συσκευές με τη χρήση του κινητού ή του tablet, απομακρυσμένα [12].
- **Εξοικονόμηση ενέργειας και πόρων**: Μπορεί να βοηθήσει στην παρακολούθηση και τη μείωση της κατανάλωσης ενέργειας, τη βελτιστοποίηση της χρήσης πόρων και την ενίσχυση της βιωσιμότητας. Δεν είναι τυχαίο άλλωστε ότι εργοστάσια παραγωγής ηλεκτρικής ενέργειας, παρακολουθούν σε καθημερινή βάση την ενεργειακή κατανάλωση του δικτύου αλλά και των καταναλωτών, σε πραγματικό χρόνο [13].
- **Περιπτώσεις έκτακτης ανάγκης**: Σε περίπτωση έκτακτης ανάγκης, υπάρχουν αισθητήρες οι οποίοι μπορούν να ανιχνεύσουν κάθε πιθανό κίνδυνο και να προειδοποιήσουν έγκαιρα σε φορείς και πολίτες. Για παράδειγμα, υπάρχουν αισθητήρες που εντοπίζουν ένα τροχαίο ατύχημα ή μία εκδήλωση πυρκαγιάς σε δάσος, και προβαίνουν άμεσα σε ειδοποίηση των αρμόδιων αρχών, ώστε να μη χαθεί πολύτιμος χρόνος.

1.4. Μειονεκτήματα Χρήσης IoT

Όπως κάθε τεχνολογία εκτός από τα οφέλη που προσφέρει, διαθέτει και ορισμένα μειονεκτήματα. Αυτά είναι τα παρακάτω:

- **Ασφάλεια και προστασία προσωπικών δεδομένων**: Όσο αυξάνεται ο αριθμός των συνδεδεμένων συσκευών τόσο αυξάνεται ο κίνδυνος για παραβίαση και υποκλοπή προσωπικών δεδομένων. Οι συνδεδεμένες συσκευές είναι ευάλωτες σε επιθέσεις στον κυβερνοχώρο και σε παραβιάσεις δεδομένων [14] [15].

- **Ανεργία**: Οι ανειδίκευτοι κυρίως εργαζόμενοι αλλά όχι μόνο, διατρέχουν τον κίνδυνο να χάσουν τη δουλειά τους, οδηγώντας σε υψηλά ποσοστά ανεργίας. Τα ρομπότ, οι και οι αυτοματοποιημένες διαδικασίες, αντικαθιστούν τους ανθρώπους που παλαιότερα έκαναν τις συγκεκριμένες εργασίες.
- **Πολυπλοκότητα**: Η σχεδίαση, η ανάπτυξη, η συντήρηση και η ενεργοποίηση της συγκεκριμένης τεχνολογίας IoT είναι αρκετά περίπλοκη και απαιτείται ειδικός τεχνικός. Ακόμα, επειδή τα συγκεκριμένα συστήματα είναι πολύπλοκα, υπάρχουν περισσότερες πιθανότητες αποτυχίας. Επιπλέον,
- **Εξάρτηση από τη χρήση του διαδικτύου**: Η επικοινωνία των συσκευών και γενικότερα του Internet of Things, βασίζονται σε μεγάλο βαθμό στο διαδίκτυο και δεν μπορούν να λειτουργήσουν αποτελεσματικά χωρίς αυτό.
- **Περιορισμένη πνευματική και σωματική δραστηριότητα**: Η υπερβολική χρήση του διαδικτύου και της τεχνολογίας καθιστά τους ανθρώπους αδρανείς, επειδή εξαρτώνται σε μεγάλο βαθμό από τις έξυπνες συσκευές. Αυτό έχει σαν αποτέλεσμα οι ανθρώπου να μην έχουν κίνητρα για να ασχοληθούν με την καλλιέργεια και φροντίδα του εαυτού τους, καθώς όλες οι εργασίες στο περιβάλλον που δραστηριοποιούνται, πραγματοποιούνται αυτόματα από τις έξυπνες συσκευές.

1.5. Βιβλιογραφική έρευνα

1.5.1. Εισαγωγή

Η έρευνα που πραγματοποιήθηκε στην διεθνή βιβλιογραφία αναφέρεται στη χρήση του Internet of Things (IoT) σχετικά με την εξοικονόμηση ενέργειας, τη μείωση του κόστους και τη βελτίωση της ποιότητας ζωής του ανθρώπου, παράλληλα με την προστασία του περιβάλλοντος. Το Internet of Things αναφέρεται στο δίκτυο συνδεδεμένων συσκευών και αισθητήρων που μπορούν να επικοινωνούν μεταξύ τους και να ανταλλάσσουν δεδομένα.

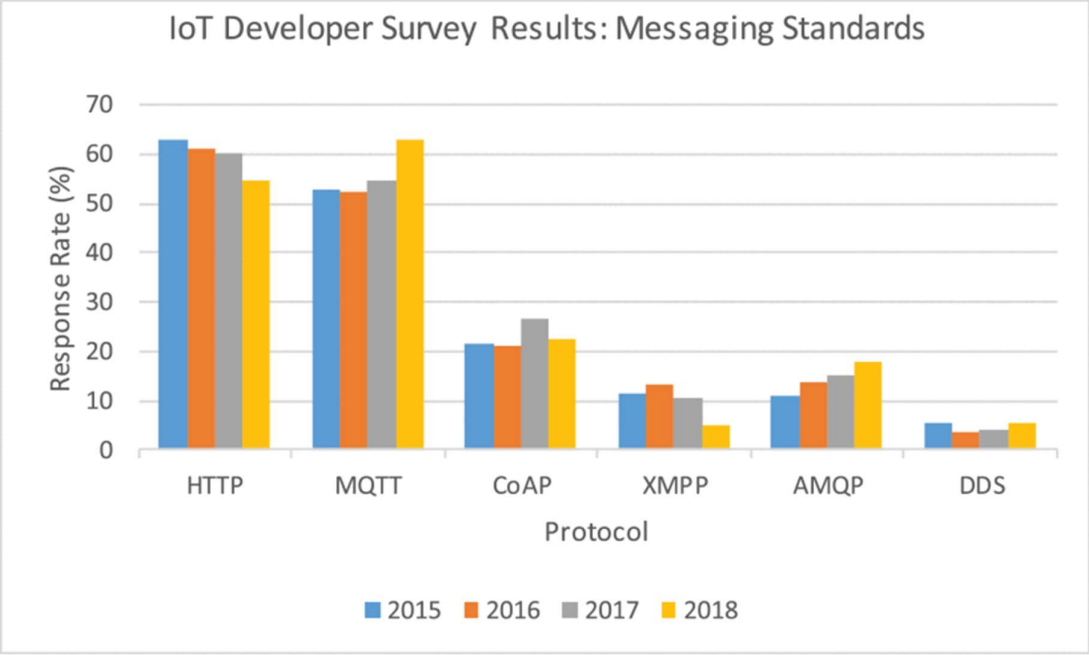
Η ανάπτυξη ενός συστήματος παρακολούθησης της ενέργειας σε πραγματικό χρόνο με τη χρήση του IoT μπορεί να αποφέρει πολλά οφέλη όσον αφορά την ενεργειακή απόδοση, την εξοικονόμηση κόστους και τη προστασία του περιβάλλοντος. Με τη χρήση αισθητήρων και έξυπνων μετρητών, μπορούν να συλλέγονται δεδομένα και να καταγράφεται η κατανάλωση ενέργειας σε κτίρια και οικιακές συσκευές. Αυτά τα δεδομένα μπορούν να αναλυθούν και να πληροφορήσουν το χρήστη, για πιθανές αστοχίες που επιφέρουν αύξηση στην κατανάλωση ενέργειας αλλά και του κόστους [16].

1.5.2. Έρευνα σε πρωτόκολλα επικοινωνίας

Καθώς ο αριθμός των συσκευών πολλαπλασιάζεται, το μέγεθος και η ταχύτητα των δεδομένων αυξάνεται. Τα συστήματα IoT βασίζονται κυρίως στη χρήση πρωτοκόλλων ανταλλαγής μηνυμάτων, για την ανταλλαγή δεδομένων IoT και υπάρχουν διάφορα πρωτόκολλα ή πλαίσια που υποστηρίζουν διαφορετικούς τύπους προτύπων ανταλλαγής μηνυμάτων. Δεδομένου ότι οι συσκευές IoT διαθέτουν συνήθως περιορισμένους υπολογιστικούς πόρους και επεξεργαστική ισχύ, η επιλογή ενός ελαφρού, αξιόπιστου, κλιμακούμενου, διαλειτουργικού, επεκτάσιμου και ασφαλούς πρωτοκόλλου ανταλλαγής μηνυμάτων καθίσταται μία δύσκολη απόφαση [17]. Συνεπώς, δεν είναι ασυνήθιστο τα συστήματα IoT να χρησιμοποιούν πολλαπλά πρότυπα ανταλλαγής μηνυμάτων για την υποστήριξη της ετερογένειας των συσκευών. Η αναφορά των πρωτοκόλλων με τα πλεονεκτήματα και τις αδυναμίες τους παρουσιάζεται στο δεύτερο κεφάλαιο.

Σε αυτό το σημείο θα παρουσιάσουμε τα αποτελέσματα μιας ετήσιας έρευνας που διεξήχθη από το Eclipse Foundation και την Eclipse IoT Working Group μεταξύ 2015 και 2018. Η έρευνα περιείχε μια σειρά ερωτήσεων προς τους προγραμματιστές και τους σχεδιαστές εφαρμογών. Επιλέξαμε να παρουσιάσουμε τα αποτελέσματα στην ερώτηση: "Ποια

πρωτόκολλα ανταλλαγής μηνυμάτων χρησιμοποιείτε για τη λύση IoT που αναπτύσσετε;". Το ποσοστό ανταπόκρισης στην απάντηση αυτής της ερώτησης για τις έρευνες που πραγματοποιήθηκαν μεταξύ 2015 και 2018 παρουσιάζεται στην παρακάτω εικόνα. Παρατηρούμε ότι τα πρωτόκολλα MQTT και AMQP έχουν αυξητικές τάσεις τα τελευταία χρόνια.



Εικόνα 1 Χρήση πρωτοκόλλων από προγραμματιστές από 2015 έως 2018

1.5.3. Πρωτόκολλα υλοποίησης εργασίας

Η επιλογή του κατάλληλου πρωτοκόλλου αποτελεί θεμέλιο για την αποτελεσματική λειτουργία ολόκληρου του συστήματος IoT και της εφαρμογής που καλείται να κατασκευάσει ο προγραμματιστής, έτσι ώστε να είναι εύκολα επεκτάσιμο.

Η επιλογή των πρωτοκόλλων AMQP και MQTT για την υλοποίηση της συγκεκριμένης εργασίας έγινε για τους παρακάτω λόγους.

- **Επεκτασιμότητα**: Το πρωτόκολλο AMQP είναι αρκετά επεκτάσιμο πρωτόκολλο, επειδή διαθέτει queues και topics για την οργάνωση της πληροφορίας. Αυτό σημαίνει ότι στην εφαρμογή που δημιουργήσαμε και αντλήσαμε δεδομένα από δύο αισθητήρες, θα υπάρχει η δυνατότητα στο μέλλον να επεκτείνουμε τα queues και τα αντίστοιχα topics έτσι ώστε να ομαδοποιήσουμε την πληροφορία και να συμπεριλάβουμε πληθώρα νέων μετρητών.
- **Ασφάλεια**: Το AMQP προσφέρει μεγαλύτερη ασφάλεια σε σχέση με το MQTT. Για το συγκεκριμένο λόγο θα χρησιμοποιήσουμε το πρωτόκολλο AMQP για να στέλνουμε τα δεδομένα τα οποία θα αποθηκεύεται σε ουρές, ενώ με το πρωτόκολλο MQTT θα λαμβάνουν την πληροφορία οι εξωτερικές εφαρμογές που έχουν κάνει subscribe στα κατάλληλο topics.
- **Εμπιστοσύνη**: Το πρωτόκολλο MQTT το εμπιστεύονται κολοσσοί στο χώρο της τεχνολογίας και του Internet of Things όπως η είναι η Amazon και το Facebook.
- **Ελαφρύ και αποδοτικό**: Το MQTT έχει σχεδιαστεί για να είναι ελαφρύ και αποδοτικό, καθιστώντας το κατάλληλο για καταστάσεις όπου το εύρος ζώνης του δικτύου είναι περιορισμένο.
- **Offline ανάκτηση μηνυμάτων:** Με το πρωτόκολλο AMQP, οι πελάτες μπορούν να ανακτούν δεδομένα όταν βρίσκονται εκτός σύνδεσης
- **Cloud**: Το MQTT είναι μια δημοφιλής επιλογή για εφαρμογές IoT που βασίζονται στο cloud. Η εφαρμογή που έχουμε κατασκευάσει θα πρέπει να φιλοξενηθεί στο cloud έτσι ώστε να είναι εκτελείται συνεχώς.

1.5.4. Επιλεγμένοι αισθητήρες υλοποίησης εργασίας

Στόχος της εργασίας αποτελεί η εξοικονόμηση ενέργειας, η μείωση του κόστους και η βελτίωση της ποιότητας ζωής του ανθρώπου, προστατεύοντας συγχρόνως και το περιβάλλον. Για να το πετύχουμε τον παραπάνω στόχο, δημιουργήθηκε εφαρμογή που θα αναφέρουμε παρακάτω, με σκοπό να αντλεί δεδομένα από τους αισθητήρες, να τα καταγράφει και τέλος να απεικονίζει τις μετρήσεις με εποπτικό τρόπο. Έπειτα, καθώς έχουμε διαθέσιμη την πληροφορία, δημιουργούνται συγκεκριμένοι κανόνες οι οποίοι αναλύουν τα δεδομένα και μόλις παρατηρήσουν απόκλιση από τις συνθήκες που έχουμε δημιουργήσει, προβαίνει σε κατάλληλες συνθήκες ώστε να πετύχουμε βελτιστοποίηση των πόρων και εξοικονόμηση ενέργειας.

Για την υλοποίηση της εφαρμογής, επιλέξαμε δύο αισθητήρες με βάση τη διαθεσιμότητα στην αγορά, την ακρίβεια των μετρήσεων, την πληθώρα συνδέσεων καθώς και το κόστος. Οι δύο μετρητές που επιλέξαμε είναι ο μετρητής ενέργειας “Janitza D21” και ο μετρητής περιβάλλοντος “Bosch BME280”. Η επιλογή των αισθητήρων πραγματοποιήθηκε επειδή καλύπτουν τα παραπάνω χαρακτηριστικά που αναφέραμε. Οι μετρήσεις που θέλουμε να αντλήσουμε αφορούν σε οικιακό και όχι εμπορικό περιβάλλον, επομένως και οι δύο είναι αρκετά ακριβείς για τη χρήση που προορίζονται. Επιπλέον, ο μετρητής περιβάλλοντος είναι αρκετά φθηνός (19,34€), ενώ λίγο ακριβότερος είναι ο μετρητής ενέργειας (80,60€). Να σημειώσουμε εδώ ότι σε ορισμένα καταστήματα ενδέχεται να μην υπάρχει ο μετρητής ενέργειας ετοιμοπαράδοτος και να χρειάζεται παραγγελία από την εταιρεία. Σε κάθε περίπτωση μέσα σε ελάχιστες ημέρες θα είναι έτοιμος προς παράδοση στο χώρο σας για να τον εγκαταστήσετε.

Στα επόμενα κεφάλαια αναλύονται διεξοδικά ο τρόπος σύνδεσης των αισθητήρων καθώς και η εφαρμογή που δημιουργήσαμε για να αντλεί, να μεταφέρει και να αποθηκεύει τα δεδομένα, με σκοπό να τα παρουσιάζει με εποπτικό τρόπο στον τελικό χρήστη.

Κεφάλαιο 2. Έξυπνη πόλη & Πρωτόκολλα επικοινωνίας IoT

2.1. Έξυπνη πόλη (Smart city)

Είναι σαφές ότι ζούμε σε μια εποχή τεχνολογικής επανάστασης, η οποία αποσκοπεί στη βελτίωση και τη διευκόλυνση της καθημερινής μας ζωής. Μια επανάσταση που ξεκίνησε πριν από περίπου μια δεκαετία, με την έλευση του έξυπνου κινητού και εξελίχθηκε με τη αποθήκευση των πληροφοριών στο σύννεφο. Στις ημέρες που διανύουμε, το Διαδικτύου των Αντικειμένων είναι πρωταγωνιστής, επειδή είναι σε θέση να διασυνδέσει όλες τις συσκευές με το Διαδίκτυο.

Μια έξυπνη πόλη είναι μια αστική περιοχή που χρησιμοποιεί ποικιλία καινοτόμων τεχνολογιών από αισθητήρες, για τη συλλογή δεδομένων. Τα δεδομένα αυτά επεξεργάζονται από λογισμικό με απώτερο στόχο την αποτελεσματική διαχείριση των πόρων και των υπηρεσιών της πόλης [20]. Η έξυπνη τεχνολογία επιτρέπει στους διαχειριστές της πόλης, έχοντας εικόνα των νευραλγικών υποδομών της, να αλληλοεπιδρούν άμεσα με αυτές, και να βελτιστοποιούν την αποτελεσματικότητα των λειτουργιών και των υπηρεσιών της, σε ταυτόχρονη διασύνδεση με τους πολίτες της. Η σχέση των πολιτών με τους φορείς της πόλης γίνεται αμφίδρομη και διαδραστική μέσω καινοτόμων επικοινωνιακών διαύλων. Τα δεδομένα που συλλέγονται χρησιμοποιούνται για την αύξηση του αριθμού των υπηρεσιών προς τους πολίτες και τη βελτίωση της ποιότητας των υπηρεσιών, με παράλληλη μείωση του λειτουργικού κόστους.

Η έξυπνη πόλη παρακολουθεί, συλλέγει και χρησιμοποιεί δεδομένα για τη διαχείριση της κυκλοφορίας και των συστημάτων μεταφορών, των σταθμών παραγωγής ενέργειας, των υπηρεσιών κοινής ωφέλειας, των δικτύων ύδρευσης, των αποβλήτων, των σχολείων, των βιβλιοθηκών και των νοσοκομείων σε πραγματικό χρόνο.

Τα βασικά πλεονεκτήματα μίας έξυπνης πόλης είναι τα εξής:

- Βελτίωση της ποιότητα ζωής κατοίκων – επισκεπτών
- Αμφίδρομη επικοινωνία μεταξύ πολιτών και φορέων
- Αυξημένη ασφάλεια
- Βέλτιστη διαχείριση πόρων
- Εξοικονόμηση ενέργειας
- Βιώσιμη λειτουργία του κράτους και των υποδομών
- Προστασία του περιβάλλοντος
- Μείωση λειτουργικού κόστους

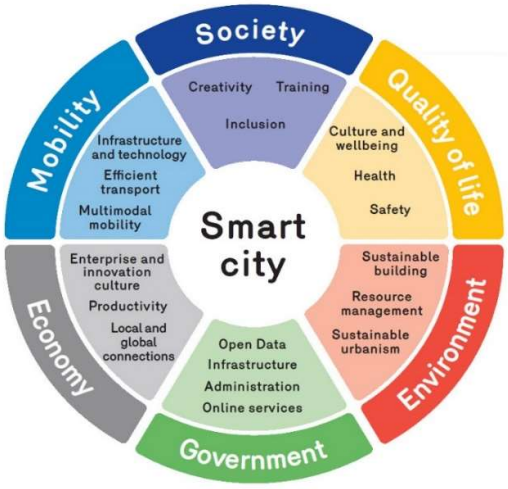
Εικόνα 2 Αισθητήρες σε μία έξυπνη πόλη

2.2. Βασικά χαρακτηριστικά έξυπνης πόλης

Σε μια έξυπνη πόλη, τα συστήματα μεταφοράς, η πρόσβαση στην υγειονομική περίθαλψη, η εκπαίδευση, οι ενεργειακές υποδομές και η διαχείριση αποβλήτων είναι συνδεδεμένα και αλληλεπιδρούν, με σκοπό να δημιουργήσουν ένα πιο αποτελεσματικό και βιώσιμο περιβάλλον.

Παρακάτω περιγράφονται τα χαρακτηριστικά μίας έξυπνης πόλης.

- **Έξυπνη οικονομία:** Με τον συγκεκριμένο όρο αναφερόμαστε στις δυνατότητες που μας παρέχει η τεχνολογία, προκειμένου να αυξηθεί η παραγωγικότητα, να ενισχυθεί η ανταγωνιστικότητα και να δημιουργηθούν νέα προϊόντα που θα εξυπηρετούν, βασικές ανάγκες των πολιτών. Οι εταιρείες προκειμένου να αυξήσουν τον ανταγωνισμό, αξιοποιούν την τεχνολογία με σκοπό να επιδιώξουν την ταχύτερη παραγωγή αγαθών με το μικρότερο δυνατό κόστος [21].



Εικόνα 3 Βασικά χαρακτηριστικά έξυπνης πόλης

- **Έξυπνη διακυβέρνηση:** Αναφέρεται στη δυνατότητα παροχής ψηφιακών υπηρεσιών από τους δημόσιους φορείς προς τους πολίτες. Το κράτος ψηφιοποιεί τις διαδικασίες με σκοπό να εξαλείψει τη μετακίνηση, την αναμονή σε ουρές και τη γραφειοκρατία και να προσφέρει εξατομικευμένες υπηρεσίες στους πολίτες δίχως χρονικούς και γεωγραφικούς περιορισμούς [22].

- **Κοινωνία:** Μία πόλη για να χαρακτηριστεί ως έξυπνη θα πρέπει και οι πολίτες που την αποτελούν να χαρακτηρίζονται ως ενεργοί. Οι πολίτες, επιδιώκουν να είναι συνεχώς ενημερωμένοι, να εξελίσσονται και να καινοτομούν, όχι μόνο για τον εαυτό τους αλλά και για το κοινωνικό σύνολο. Ειδικά στην εποχή μας, η καινοτομία και η δημιουργικότητα αποτελούν πρωταρχικό παράγοντα για την πρόοδο και την ευημερία των κοινωνιών [23].

- **Έξυπνοι τρόποι μετακίνησης**: Είναι γεγονός ότι η πρόοδος της τεχνολογίας έχει επηρεάσει σε μεγάλο βαθμό τον τομέα των μεταφορών, όπου αναμένεται να συνεχιστεί τα επόμενα χρόνια. Η τεχνολογία έχει εισβάλει στα σύγχρονα αυτοκίνητα αλλά και στα μέσα μαζικής μεταφοράς. Ο στόχος είναι να εκμεταλλευτούμε τα οφέλη που προσφέρονται, έτσι ώστε να πετύχουμε τη μείωση των τροχαίων ατυχημάτων, χάρη στην αμφίδρομη επικοινωνία των αυτοκινήτων και την άμεση λήψη αποφάσεων. Με αυτό τον τρόπο, θα αυξηθεί η ασφάλεια των μετακινήσεων, θα επιτευχθεί η εξοικονόμηση ενέργειας και η μείωση των εκπομπών CO2, βοηθώντας στην προστασία του περιβάλλοντος [24].

Εικόνα 4 Έξυπνη μετακίνηση

- **Έξυπνος τρόπος ζωής:** Αφορά τον τρόπο όπου το κράτος οργανώνει τις δημόσιες με τέτοιο τρόπο ώστε να βελτιώνουν τον τρόπο ζωής των πολιτών. Κυρίως αφορά την βελτίωση των υποδομών υγείας, πολιτισμού, υπηρεσίες εξυπηρέτησης πολιτών, ασφάλειας και τουρισμού [25]. Με τη χρήση της τεχνολογίας βελτιώνεται η πρόσβαση στην υγειονομική περίθαλψη, με εφαρμογές που επιτρέπουν στους πολίτες να παρακολουθούν, καταγράφουν και να στέλνουν τα δεδομένα στον ιατρό τους. Αντίστοιχα, μέσω εφαρμογών μπορεί ο πολίτης να συνομιλήσει με την κατάλληλη υπηρεσία που επιθυμεί αλλά και ο τουρίστας να λάβει όλες τις απαραίτητες πληροφορίες που επιθυμεί μέσα από μία εφαρμογή.

Εικόνα 5 Έξυπνος τρόπος ζωής

- **Περιβάλλον:** Με τα οφέλη και τη πρόοδο της τεχνολογίας, μπορούμε να πετύχουμε την εξοικονόμηση των φυσικών πόρων, την προστασία του περιβάλλοντος, τη βέλτιστη χρήση της ενέργειας καθώς και την αύξηση χρήσης ανανεώσιμων πηγών ενέργειας [26].

2.3. Πρωτόκολλα επικοινωνίας IoT

Για να μπορέσουν να επικοινωνήσουν και να ανταλλάξουν δεδομένα, οι συσκευές, θα πρέπει να χρησιμοποιούν ένα κοινό πρότυπο επικοινωνίας, με σκοπό την επίτευξη επικοινωνίας μεταξύ τους. Η επικοινωνία αυτή πραγματοποιείται χρησιμοποιώντας πρωτόκολλα επικοινωνίας.

Τα πρωτόκολλα επικοινωνίας υποδηλώνουν τις διαδικασίες και τους κανόνες που επιτρέπουν στις συσκευές IoT να αλληλεπιδρούν μεταξύ τους και να συνδέονται στο Διαδίκτυο για να μεταφέρουν δεδομένα. Είναι φανερό ότι δίχως τη χρήση των πρωτοκόλλων επικοινωνίας, θα ήταν ανέφικτη η πληθώρα των εφαρμογών, που υπάρχουν στην καθημερινή μας ζωή.

Ορισμένα από τα πιο διαδεδομένα πρωτόκολλα επικοινωνίας IoT περιλαμβάνουν το HTTP, το MQTT, το CoAP, το AMQP κ.ά. Κάθε ένα από αυτά χρησιμοποιείται για διαφορετικές εφαρμογές, ενώ η επιλογή του κατάλληλου πρωτοκόλλου εξαρτάται από την εφαρμογή, την ασφάλεια, την εμβέλεια και το σενάριο χρήσης.

Υπάρχει πληθώρα πρωτοκόλλων επικοινωνίας, με σκοπό να καλυφθούν διαφορετικά σενάρια χρήσης, με τα ανάλογα πλεονεκτήματα και μειονεκτήματα.

Παρακάτω, θα αναλυθούν τα πρωτόκολλα και τα χαρακτηριστικά τους.

2.3.1. Wi-Fi

Το πρωτόκολλο Wi-Fi είναι ένα πρότυπο επικοινωνίας που χρησιμοποιείται για την ασύρματη διασύνδεση συσκευών σε ένα δίκτυο, συμπεριλαμβανομένων υπολογιστών, smartphones, tablets και άλλων συσκευών. Αποτελεί μία ευρέως διαδεδομένη και αποδοτική τεχνολογία ασύρματης επικοινωνίας, επιτρέποντας την σύνδεση των συσκευών τόσο με το διαδίκτυο όσο και με το τοπικό δίκτυο.

Το Wi-Fi αναπτύχθηκε και κυκλοφόρησε για πρώτη φορά το 1997, υπό το σύνολο των κανόνων του προτύπου IEEE 802.11. Το IEEE 802.11 είναι ένα σύνολο προτύπων για την επικοινωνία μέσω ασύρματου τοπικού δικτύου (WLAN) που θεσπίστηκε από το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE). Διαφορετικές εκδόσεις του προτύπου 802.11 έχουν κυκλοφορήσει με την πάροδο του χρόνου, κάθε μία από τις οποίες βελτιώνει και επαυξάνει την προηγούμενη, όσον αφορά την ταχύτητα, την εμβέλεια και τη χωρητικότητα.

Το πρότυπο συνεχίζει να επεκτείνεται συνεχώς, με σκοπό να προστεθούν νέες δυνατότητες και να διορθωθούν οι αδυναμίες του.

Τα βασικά πλεονεκτήματα που το καθιστούν ευρέως διαδεδομένο είναι η δυνατότητα ασύρματης σύνδεσης χωρίς τη χρήση καλωδίων, η ευκολία χρήσης ακόμα και από χρήστες με ελάχιστη εμπειρία, η συμβατότητα σύνδεσης με πληθώρα συσκευών, η υποστήριξη πολλαπλών συσκευών και χρηστών στο ίδιο δίκτυο αλλά και η αυξημένη ταχύτητα μετάδοσης και μεταφοράς δεδομένων, ειδικά στις τελευταίες εκδόσεις του προτύπου [27].

Από την άλλη, τα βασικά μειονεκτήματα είναι η περιορισμένη εμβέλεια όταν υπάρχουν εμπόδια μεταξύ του δρομολογητή και της συσκευής και οι παρεμβολές από άλλα δίκτυα στην ίδια συχνότητα εκπομπής, οι οποίες επιφέρουν κατακόρυφη πτώση στην απόδοση του δικτύου. Το πιο σημαντικό μειονέκτημα όμως εξακολουθεί να είναι η έλλειψή ασφάλειας σε σύγκριση με ένα ασύρματο δίκτυο. Παρόλο την πρόοδο της κρυπτογράφησης, η επικοινωνία μέσω Wi-Fi παραμένει πιο ευάλωτη σε επιθέσεις σε σχέση με ένα ενσύρματο δίκτυο.

Εικόνα 6 Λογότυπο ασύρματης επικοινωνίας Wi-Fi

2.3.2. Bluetooth

Το Bluetooth είναι ένα πρωτόκολλο ασύρματης επικοινωνίας μικρής εμβέλειας που χρησιμοποιείται για τη σύνδεση των συσκευών και τη μεταφορά των δεδομένων σε κοντινή απόσταση. Είναι ένα τυποποιημένο πρωτόκολλο για την αποστολή και λήψη δεδομένων μέσω ασύρματης σύνδεσης 2,4 GHz. Πρόκειται για ένα ασφαλές πρωτόκολλο και είναι ιδανικό για ασύρματες μεταδόσεις μικρής εμβέλειας, χαμηλής ισχύος και χαμηλού κόστους μεταξύ ηλεκτρονικών συσκευών.

Το Bluetooth ξεκίνησε το 1998 με την ίδρυση του Bluetooth Special Interest Group (SIG), μίας μη κερδοσκοπικής ένωσης που προέκυψε από τη σύμπραξη πέντε εταιρειών: Ericsson, IBM, Intel, Nokia και Toshiba. Το πρώτο εμπορικό προϊόν παρουσιάστηκε ένα χρόνο αργότερα. Χάρη στην ευκολία, την ευελιξία και το χαμηλό κόστος, το Bluetooth έγινε ευρέως γνωστό, επειδή υιοθετήθηκε από πληθώρα προϊόντων όπως κινητά τηλέφωνα και υπολογιστές. Αποτελεί επίσης βασικό χαρακτηριστικό σε έξυπνους αισθητήρες για τη μεταφορά των δεδομένων σε μία έξυπνη πόλη.

Το πρότυπο IEEE 802.15.1 είναι ένα πρότυπο ασύρματης επικοινωνίας που καθιερώθηκε από το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE) και είχε ως στόχο να δημιουργήσει μια μέθοδο για την παροχή ασύρματης σύνδεσης μικρής εμβέλειας, χαμηλής κατανάλωσης ενέργειας και ασφαλούς σύνδεσης μεταξύ ενός ευρέος φάσματος συσκευών. Το πρότυπο επικοινωνίας δημιουργήθηκε από την έκδοση Bluetooth 1.1. Το πρότυπο εμπίπτει στην ευρύτερη κατηγορία του IEEE 802.15, το οποίο είναι το τμήμα του πρωτοκόλλου IEEE 802 που διέπει τα πρότυπα ασύρματων προσωπικών δικτύων (Wireless Personal Area Network - WPAN).

Τα βασικά πλεονεκτήματα που το καθιστούν ευρέως διαδεδομένο είναι η δυνατότητα ασύρματης σύνδεσης χωρίς τη χρήση καλωδίων, η συμβατότητα σύνδεσης με πληθώρα συσκευών και η χαμηλή κατανάλωση ενέργειας, που έχει συμπεριληφθεί από την έκδοση 4.0 του προτύπου, προσφέροντας σημαντικές βελτιώσεις στην ενεργειακή απόδοση. Συνεπώς, καθίσταται ιδιαίτερα φιλικό πρότυπο για να συμπεριληφθεί σε πληθώρα αισθητήρων που συμπεριλαμβάνονται από τα έξυπνα κινητά τηλέφωνα μέχρι εφαρμογές διαχείρισης μίας έξυπνης πόλης [28].

Από την άλλη, τα βασικά μειονεκτήματα είναι το περιορισμένο εύρος σύνδεσης σε σύγκριση με το Wi-Fi, το οποίο μπορεί να αποτελεί πρόβλημα σε συγκεκριμένες εφαρμογές, οι παρεμβολές από άλλα δίκτυα στην ίδια συχνότητα εκπομπής, οι οποίες μπορεί να οδηγήσουν ακόμα και σε αδυναμία σύνδεσης, η αργή ταχύτητα μετάδοσης δεδομένων σε σύγκριση με τις ταχύτητες μετάδοσης δεδομένων του Wi-Fi και ο περιορισμένος αριθμός συνδεδεμένων συσκευών ταυτόχρονα. Το συγκεκριμένο μειονέκτημα περιορίζει σε μεγάλο βαθμό τις συσκευές που μπορούν να αλληλοεπιδράσουν με άλλες συσκευές ταυτόχρονα, σε σχέση με το πρωτόκολλο επικοινωνίας Wi-Fi που γνωρίσαμε προηγουμένως.

Εικόνα 7 Λογότυπο Bluetooth

2.3.3. Zigbee

Το Zigbee είναι ένα ασύρματο πρωτόκολλο επικοινωνίας που βασίζεται στο πρότυπο IEEE 802.15.4 για δίκτυα χαμηλής ταχύτητας και ελάχιστη κατανάλωση ενέργειας. Συχνά χρησιμοποιείται σε εφαρμογές Internet of Things (IoT) για αυτοματισμούς σπιτιών.

Το Zigbee είναι ένα πρωτόκολλο που χρησιμοποιείται για ασύρματη επικοινωνία μεταξύ συσκευών. Το πρότυπο δημιουργήθηκε το 2002 από την ZigBee Alliance. Περιγράφεται στο πρότυπο IEEE 802.15.4 και λειτουργεί στις ζώνες 868 MHz, 902-928 MHz και 2,4 GHz. Ανήκει στο PAN (Personal Area Network), παρόμοιο με τη γνωστή τεχνολογία Bluetooth. Είναι ένα ανοιχτό πρότυπο που εστιάζει στην επικοινωνία χαμηλής ενέργειας και χαμηλού κόστους [29].

Τα βασικά πλεονεκτήματα που το καθιστούν ευρέως διαδεδομένο είναι η υψηλή ασφάλεια καθώς περιλαμβάνει πολλαπλά επίπεδα ασφάλειας, συμπεριλαμβανομένης της κρυπτογράφησης, για να προστατεύσει τα δεδομένα. Επιπλέον, ένα ακόμη χαρακτηριστικό που διαθέτουν είναι η χαμηλή κατανάλωση ενέργειας. Οι συσκευές Zigbee είναι σχεδιασμένες για να καταναλώνουν ελάχιστη ενέργεια, γεγονός που τις καθιστά ιδανικές για μακροχρόνια χρήση χωρίς να απαιτείται αντικατάσταση ή φόρτιση της μπαταρίας. Από την άλλη μεριά, σε σχέση με άλλα πρότυπα όπως το Wi-Fi, το Zigbee προσφέρει σχετικά χαμηλές ταχύτητες μετάδοσης δεδομένων. Αυτό μπορεί να μη θεωρείται πρόβλημα, για εφαρμογές όπως οι αισθητήρες που μεταφέρουν μικρές ποσότητες δεδομένων, αλλά μπορεί να είναι ακατάλληλο για εφαρμογές που απαιτούν μεταφορά μεγαλύτερου όγκου δεδομένων. Αν και το Zigbee είναι ένα ανοιχτό πρότυπο, υπάρχουν πολλές διαφορετικές εκδόσεις που ενδέχεται να μην είναι συμβατές μεταξύ τους. Αυτό μπορεί να δημιουργήσει προβλήματα στη σύνδεση συσκευών από διαφορετικούς κατασκευαστές. Τέλος, η εμβέλεια που μπορεί να καλύψει μια μεμονωμένη συσκευή Zigbee είναι σχετικά μικρή, ειδικά σε σύγκριση με τεχνολογίες όπως το Wi-Fi. Αυτό σημαίνει ότι οι συγκεκριμένες συσκευές, ενδέχεται να εμφανίσουν πρόβλημα στη σύνδεση όταν υπάρχουν εμπόδια τα οποία εξασθενούν το σήμα.

Εικόνα 8 Λογότυπο Zigbee

2.3.4. MQTT

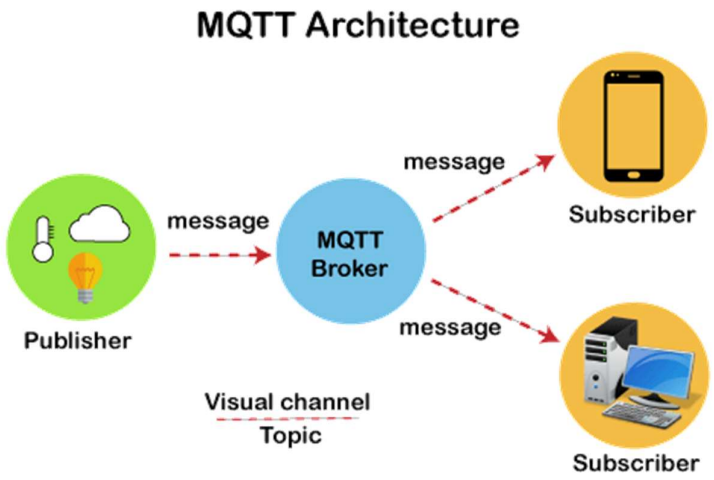
Εικόνα 9Λογότυπο MQTT

Το MQTT (Message Queuing Telemetry Transport) είναι ένα ελαφρύ πρωτόκολλο δημοσίευσης-εγγραφής (publish-subscribe) που χρησιμοποιείται για την αποστολή μηνυμάτων μεταξύ συσκευών και συχνά χρησιμοποιείται σε συστήματα Internet of Things (IoT). Το πρωτόκολλο εφευρέθηκε το 1999 από τους Andy Stanford-Clark και Arlen Nipper των IBM και Cirrus Link αντίστοιχα. Ο στόχος ήταν να δημιουργηθεί ένα πρωτόκολλο επικοινωνίας χαμηλών απαιτήσεων σε ενέργεια και δικτυακό εύρος (bandwidth) ώστε να παρακολουθείται και να συντονίζεται η λειτουργία αγωγών πετρελαίου δια μέσου δορυφόρου. [30]

Ο τρόπος λειτουργίας του στηρίζεται σε δύο βασικούς όρους στο διαμεσολαβητή (broker) που είναι συνδεδεμένοι οι πελάτες (subscribers), οι οποίοι με την σειρά τους μπορεί να είναι είτε publisher (συσκευή που αναρτά ένα μήνυμα στον broker) είτε subscriber (συσκευή που εγγράφεται σε ένα θέμα στον broker). Με τον όρο broker εννοούμε τον διαμεσολαβητή ο οποίος διαχειρίζεται και μεταβιβάζει τα μηνύματα μεταξύ του εκδότη και των συνδρομητών. Από την πλευρά του ο publisher έχει την δυνατότητα να στέλνει τα αντίστοιχα μηνύματα στον broker βασισμένα πάνω στο συγκεκριμένο θέμα που έχει λάβει, ενώ αντίστοιχα ο subscriber πραγματοποιεί εγγραφή στον broker [31]. Ο broker γενικότερα γνωρίζει του πελάτες που είναι συνδεδεμένοι σε αυτόν άρα όταν θα λάβει τα αντίστοιχα μηνύματα θα έχει την δυνατότητα να τα στείλει στον αντίστοιχο subscriber [75].

Τα βασικά πλεονεκτήματα που το καθιστούν ευρέως διαδεδομένο είναι: αξιόπιστο επειδή παρέχει διάφορα επίπεδα Quality of Service (QoS), εξασφαλίζοντας ότι τα μηνύματα παραδίδονται αξιόπιστα ακόμη και σε αναξιόπιστα δίκτυα. Έχει σχεδιαστεί για να χρησιμοποιεί ελάχιστα δεδομένα και μικρή κατανάλωση ενέργειας, γεγονός που το καθιστά ιδανικό για περιορισμένα δίκτυα και συσκευές IoT με μπαταρία [32]. Επιπλέον, το μοντέλο δημοσίευσης-εγγραφής (publish - subscribe) που χρησιμοποιεί το MQTT παρέχει μεγάλη ευελιξία και επιτρέπει στις συσκευές να αλληλεπιδρούν σε πραγματικό χρόνο.

Το βασικό μειονέκτημα του συγκεκριμένου πρωτοκόλλου σχετίζεται με την ασφάλεια, καθώς δεν διαθέτει κρυπτογράφηση, στοιχείο που το καθιστά ιδιαίτερο ευάλωτο σε επιθέσεις και υποκλοπές δεδομένων.



Εικόνα 10 Βασική αρχιτεκτονική και λειτουργία του πρωτοκόλλου MQTT

2.3.5. AMQP

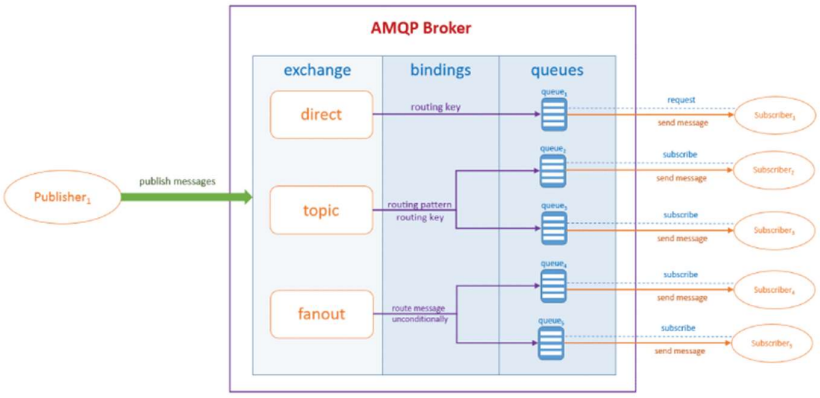
Το AMQP (Advanced Message Queuing Protocol) είναι ένα ανοικτό πρωτόκολλο επιπέδου εφαρμογής που έχει σχεδιαστεί για να επιτρέπει την επικοινωνία μεταξύ των ενδιάμεσων λογισμικών που χρησιμοποιούν μηνύματα. Η ιδέα για την υλοποίηση του AMQP προτάθηκε το 2003 από τον John O'Hara της JPMorgan Chase στο Λονδίνο. Το AMQP σχεδιάστηκε με στόχο την ασφάλεια, την αξιοπιστία, τη διαλειτουργικότητα και την ανοικτή πρόσβαση στους προγραμματιστές.

Το AMQP υποστηρίζει ένα ευρύ φάσμα εφαρμογών ανταλλαγής μηνυμάτων και προτύπων επικοινωνίας και παρέχει ελεγχόμενη επικοινωνία καθώς και επιβεβαίωση παράδοσης μηνυμάτων. Αποτελεί ένα πρωτόκολλο ανοιχτής επικοινωνίας ώστε να επιτυγχάνεται η ανταλλαγή μηνυμάτων μεταξύ εφαρμογών. Τα καθοριστικά χαρακτηριστικά του αφορούν τον προσανατολισμό, τη δρομολόγηση, την αξιοπιστία και την ασφάλεια. Χρησιμοποιείται από πολυάριθμες εφαρμογές λόγο της ευελιξίας που το διακρίνει καθώς έχει την δυνατότητα διαχείρισης μηνυμάτων ακόμα και σε μεγάλη κλίμακα όπου η επεξεργασία των μηνυμάτων αυτών πραγματοποιείται ασύγχρονα και μεταξύ ανεξάρτητων συστοιχιών.

Τα βασικά του πλεονεκτήματα που το καθιστούν ευρέως γνωστό είναι γιατί παρέχει μηχανισμούς για τη διασφάλιση της αξιοπιστίας των μηνυμάτων, συμπεριλαμβανομένων των επιβεβαιώσεων παράδοσης και της μόνιμης αποθήκευσης των μηνυμάτων [33]. Επιπλέον, το AMQP μπορεί να υποστηρίξει μια μεγάλη γκάμα εφαρμογών, από απλά σενάρια επικοινωνίας έως πολύπλοκες μεταβιβάσεις μηνυμάτων σε πληθώρα εφαρμογών. Το AMQP παρέχει ενσωματωμένα χαρακτηριστικά ασφάλειας, συμπεριλαμβανομένης της υποστήριξης για SSL/TLS καθώς και της πιστοποίησης χρηστών. Αυτό σημαίνει ότι είναι κατάλληλο πρότυπο για εφαρμογές, που η ασφάλεια κατά τη μεταφορά των δεδομένων αποτελεί τη μεγαλύτερη προϋπόθεση.

Από την άλλη πλευρά, η υλοποίηση του πρωτοκόλλου AMQP σε σχέση με το MQTT ενδέχεται να μην είναι η καλύτερη επιλογή, για τις συσκευές που διαθέτουν περιορισμένες δυνατότητες. Το AMQP απαιτεί περισσότερους πόρους τόσο σε ενέργεια όσο και σε μνήμη. Ακόμα, λόγω της πολυπλοκότητας του AMQP, απαιτείται περισσότερος χρόνος από τους προγραμματιστές για να εμβαθύνουν στη λειτουργία του σε σύγκριση με τα απλούστερα

πρωτόκολλα, όπως το MQTT. Στην παρακάτω εικόνα, μπορείτε να δείτε το μοντέλο διανομής των μηνυμάτων στο AMQP. Η συγκεκριμένη λειτουργία θα αναφερθεί στην ενότητα 3.4.1



Εικόνα 11 Μοντέλο διανομής μηνυμάτων AMQP

2.3.6. Modbus

Πρόκειται για ένα πρωτόκολλο επικοινωνίας που αναπτύχθηκε το 1979 για PLC και χρησιμοποιείται ευρέως κυρίως σε βιομηχανικές εφαρμογές. Η λειτουργία του σχετίζεται με την ανταλλαγή μηνυμάτων μεταξύ ηλεκτρικών συσκευών, χρησιμοποιώντας σειριακές γραμμές ως φυσικό μέσο.

Το Modbus είναι ένα πρωτόκολλο αίτησης-απόκρισης που υλοποιείται με τη χρήση μιας διάταξης master-slave. Σε μια σχέση master-slave, η επικοινωνία γίνεται σε ζεύγη, μια συσκευή πρέπει να ξεκινήσει ένα αίτημα και στη συνέχεια να περιμένει μια απάντηση. Για την επίτευξη της συγκεκριμένης επικοινωνίας η διάταξη master-slave θα πρέπει να συνδεθεί με ένα σειριακό καλώδιο. Στη συνέχεια, θα ξεκινήσει η διαδικασία επικοινωνίας. Τα δεδομένα αποστέλλονται ως σειρές από μονάδες και μηδενικά και ονομάζονται bits. Κάθε bit αποστέλλεται ως τάση. Τα μηδενικά αποστέλλονται ως θετικές τάσεις και οι μονάδες ως αρνητικές. Τα bits αποστέλλονται με μεγάλες ταχύτητες. Μια τυπική ταχύτητα μετάδοσης είναι 9600 baud (bits ανά δευτερόλεπτο). Με τη συγκεκριμένη διαδικασία μετάδοσης αποστέλλονται τα δεδομένα από το master στο slave.

Τα βασικά πλεονεκτήματα χρήσης του Modbus αφορούν την εύκολη υλοποίηση και χρήση του, καθώς ανήκει στην κατηγορία των ανοικτών πρωτοκόλλων και χρησιμοποιείται ελεύθερα. Επιπλέον, παρέχει πληθώρα διασύνδεσης με τα σειριακά πρωτόκολλα όπως RS-

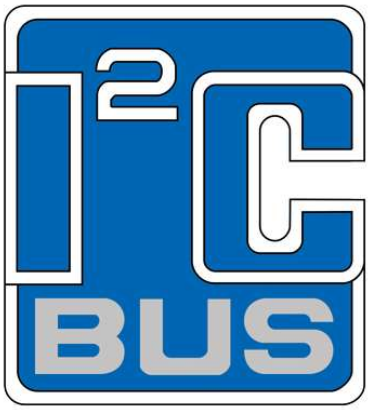
485, RS-422, RS-232 καθώς και το πρωτόκολλο επικοινωνίας TCP/IP για την επικοινωνία και τη μεταφορά δεδομένων ανάμεσα σε υπολογιστές που είναι συνδεδεμένοι σε ένα δίκτυο υπολογιστών. Από την άλλη πλευρά, το σημαντικότερο μειονέκτημα του Modbus είναι ότι δε περιλαμβάνει ασφάλεια και κρυπτογράφηση των δεδομένων κατά τη μεταφορά. Αυτό σημαίνει ότι τα δεδομένα που μεταφέρονται, επειδή δεν είναι κρυπτογραφημένα μπορεί να εκτεθούν σε τρίτους. Ακόμα, το Modbus είναι σχετικά παλιό πρωτόκολλο με αποτέλεσμα να μην προσφέρει την απόδοση, την αποδοτικότητα και τις δυνατότητες που προσφέρουν τα νεότερα [34].

Το πρωτόκολλο Modbus χωρίζεται σε δύο κατηγορίες με βάση τον τρόπο μετάδοσης: Modbus RTU και Modbus TCP/IP.

- **Modbus RTU (Remote Terminal Unit):** Σε αυτή τη λειτουργία, τα μηνύματα Modbus μεταφέρονται σε δυαδική μορφή. Κάθε byte δεδομένων σε ένα μήνυμα αποστέλλεται ως δύο δεκαεξαδικοί χαρακτήρες, βελτιώνοντας την αποδοτικότητα του πρωτοκόλλου. Το Modbus RTU χρησιμοποιείται συνήθως σε δίκτυα RS-232 ή RS-485.
- **Modbus TCP/IP**: Αυτή η λειτουργία, επίσης γνωστή ως Modbus-TCP, είναι μια έκδοση του πρωτοκόλλου Modbus που χρησιμοποιεί το TCP/IP για τη μεταφορά δεδομένων. Τα μηνύματα Modbus ενσωματώνονται σε ένα πακέτο TCP/IP, επιτρέποντάς τους να μεταδίδονται μέσω δικτύων που βασίζονται στο Ethernet και στο διαδίκτυο. Το Modbus TCP/IP διατηρεί την απλότητα του αρχικού πρωτοκόλλου Modbus, ενώ επεκτείνει την εμβέλειά του, σε συσκευές που είναι συνδεδεμένες στο διαδίκτυο.

Εικόνα 12 Λογότυπο Modbus

2.3.7. I2C

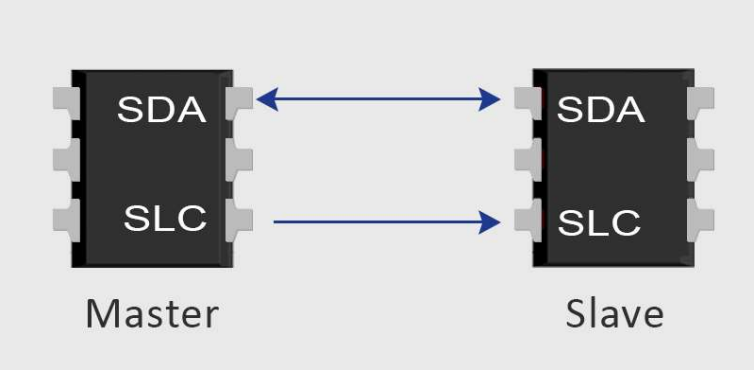


Εικόνα 13 Λογότυπο I2C

Το I²C ή I2C είναι συντομογραφία του Inter-Integrated Circuit, ένα πρωτόκολλο σειριακής επικοινωνίας που κατασκευάστηκε από την Philips Semiconductor. Δημιουργήθηκε με σκοπό την επικοινωνία μεταξύ chips που βρίσκονται στην ίδια πλακέτα τυπωμένου κυκλώματος (PCB). Συνήθως χρησιμοποιείται για τη διασύνδεση ολοκληρωμένων κυκλωμάτων με μικροεπεξεργαστή ή μικροελεγκτή. Είναι ένα πρωτόκολλο master-slave, όπου ένας επεξεργαστής ή μικροελεγκτής είναι ο master και ένας αισθητήρας ο slave. Μια συσκευή που έχει τη δυνατότητα να εκκινεί μια επικοινωνία με άλλες συσκευές σε ένα δίκτυο I2C ονομάζεται master, ενώ οι άλλες συσκευές ονομάζονται slaves. Ο master έχει την ευθύνη να εκκινεί και να τερματίζει την επικοινωνία, ενώ οι slaves απαντούν στις εντολές του master. Μπορούμε να έχουμε πολλούς master και slave στον ίδιο δίαυλο I2C. Επιπλέον, το I2C είναι ένα σειριακό πρωτόκολλο επικοινωνίας, οπότε τα δεδομένα μεταφέρονται bit προς bit κατά μήκος ενός μόνο καλωδίου (SDA). Το I2C χρησιμοποιεί μόνο δύο σειριακούς διαύλους, το SDA (Serial Data Line) για τη μεταφορά των δεδομένων και τον SCL (Serial Clock Line) για το συγχρονισμό τους. Κάθε συσκευή που συνδέεται σε ένα δίκτυο I2C έχει μια μοναδική διεύθυνση, το οποίο επιτρέπει την επικοινωνία μεταξύ του master και των συγκεκριμένων συσκευών.

Τα πλεονέκτημα χρήσης του πρωτοκόλλου I2C είναι απλό στη χρήση και στην υλοποίηση. Απαιτεί μόνο δύο γραμμές, η μία για τη μεταφορά δεδομένων (SDA) και η άλλη για το συγχρονισμό (SCL). Το I2C μπορεί να επικοινωνεί με πολλές συσκευές ταυτόχρονα. Αυτό είναι πολύ χρήσιμο για τη διασύνδεση διαφορετικών συσκευών σε ένα σύστημα. Ακόμα, έχει τη δυνατότητα να τροφοδοτήσει τις συσκευές από την ίδια γραμμή που μεταφέρει τα δεδομένα.

Από την άλλη πλευρά, τα βασικά μειονεκτήματα του πρωτοκόλλου είναι η ταχύτητα και η ασφάλεια. Η ταχύτητά του περιορίζεται συνήθως σε 3.4 Mbps, που είναι αρκετά χαμηλό για ορισμένες εφαρμογές. Ακόμα, το πρωτόκολλο δεν διαθέτει ενσωματωμένους μηχανισμούς ασφάλειας, όπως κρυπτογράφηση. Αυτό σημαίνει ότι τα δεδομένα κατά τη μεταφορά είναι εκτεθειμένα για υποκλοπή και αλλοίωση. Τέλος, ενδέχεται να παρουσιάζει ευαισθησία στις παρεμβολές σε εφαρμογές που απαιτούν μεγάλες αποστάσεις μεταξύ των συσκευών.



Εικόνα 14 Αρχιτεκτονική I2C

Κεφάλαιο 3. Αισθητήρες συλλογής μετρήσεων, εργαλεία υλοποίησης εφαρμογής API & σχεδίαση πλακέτας PCB

3.1. Εισαγωγή

Στο παρόν κεφάλαιο, θα αναλύσουμε τη λειτουργία του μετρητή ενέργειας "Janitza D21 Smart Meter Sensor", του αισθητήρα μέτρησης περιβάλλοντος "Bosch BME280 sensor", ενώ στο επόμενο θα αναλύσουμε την εφαρμογή που δημιουργήσαμε για την καταγραφή μετρήσεων από τους παραπάνω μετρητές.

Για τις ανάγκες τις διπλωματικής εργασίας χρησιμοποιήθηκε ο μετρητής ενέργειας "Janitza D21 DIN Rail Mounted Energy Meter", με τη βοήθεια του οποίου πραγματοποιήθηκε η καταγραφή των παρακάτω ενεργειακών παραμέτρων. Οι μετρήσεις που καταγράφηκαν είναι Ρεύμα (I), τάση (V), πραγματική ισχύς (P), άεργος ισχύς (Q), φαινόμενη ισχύς (S), συνφ και κατανάλωση ενέργειας.

Επιπλέον, χρησιμοποιήθηκε ο αισθητήρας μέτρησης περιβάλλοντος "Bosch BME280 sensor", με τη βοήθεια του οποίου πραγματοποιήθηκε καταγραφή θερμοκρασίας, υγρασίας και βαρομετρικής πίεσης.

Έπειτα, αναπτύχθηκε εφαρμογή σε node.js, η οποία συλλέγει τα δεδομένα από τους παραπάνω μετρητές, τα προωθεί στο RabbitMQ ενώ στο τέλος, εισάγονται στην πλατφόρμα διαχείρισης Thingsboard, χρησιμοποιώντας το πρωτόκολλο MQTT. Με το Thingsboard, πετυχαίνουμε τη μόνιμη αποθήκευση των δεδομένων που συλλέγονται από τους μετρητές, καθώς και απεικόνιση των μετρήσεων χρησιμοποιώντας γραφήματα, για την πληροφόρηση του τελικού χρήστη.

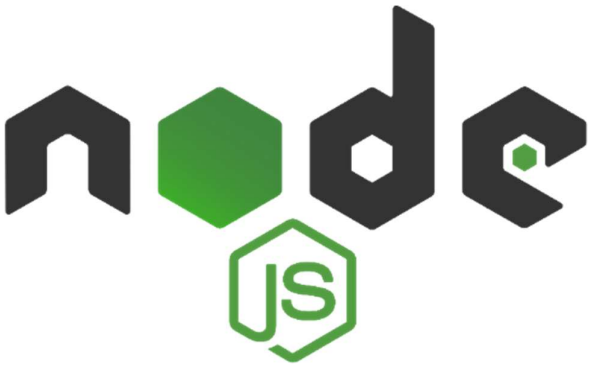
3.2. Εφαρμογή συλλογής μετρήσεων ενέργειας και περιβάλλοντος

Η εφαρμογή που αναπτύχθηκε (API), χρησιμοποιεί την πλατφόρμα ανοιχτού κώδικα Node.js. Η εφαρμογή συγκεντρώνει μετρήσεις συνθηκών περιβάλλοντος (θερμοκρασία ^{o}C, υγρασία αέρα %, βαρομετρική πίεση hPa), καθώς και μετρήσεις ενέργειας (ρεύμα, τάση, πραγματική ισχύς, άεργος ισχύς, φαινόμενη ισχύς, συνφ από τους αισθητήρες, όπως απεικονίζεται στην παρακάτω εικόνα. Στη συνέχεια, δρομολογούνται οι μετρήσεις στον message broker, ο οποιίος είναι ο RabbitMQ και στο τέλος καταλήγουν σε πλατφόρμα διαχείρισης IoT (Internet of Things) που είναι το Thingsboard και είναι ανοιχτού κώδικα.

Στο συγκεκριμένο σημείο, θα πρέπει να αναφέρουμε ότι για τη σύνδεση του μετρητή ενέργειας και του μετρητή περιβάλλοντος, χρησιμοποιήθηκε ο ελεγκτής Raspberry Pi 4. Τέλος, οι μετρήσεις του αναλυτή ενέργειας συλλέγονται ενσύρματα με χρήση του πρωτοκόλλου Modbus, ενώ οι μετρήσεις συνθηκών περιβάλλοντος συλλέγονται με τη χρήση του πρωτοκόλλου I2C (Inter-Integrated Circuit), που αναλύσαμε στο κεφάλαιο 2.

Εικόνα 15 Διάγραμμα ροής δεδομένων εφαρμογής API

3.3. Node.js



Εικόνα 16 Λογότυπο node,js

Το Node.js είναι ένα περιβάλλον εκτέλεσης JavaScript ανοιχτού κώδικα, cross-platform, που επιτρέπει στους προγραμματιστές να εκτελούν κώδικα JavaScript, χωρίς τη χρήση ενός προγράμματος περιήγησης (browser). Δημιουργήθηκε από τον Ryan Dahl το 2009 και βασίζεται στη μηχανή V8 JavaScript, η οποία αναπτύσσεται από την Google και χρησιμοποιείται στο πρόγραμμα περιήγησης ιστού Chrome [35].

3.3.1. Παράγοντες επιλογής του node.js

Η επιλογή του Node.js για τη συγγραφή της εργασίας, έγινε, επειδή το περιβάλλον προσφέρει πολλά χαρακτηριστικά και δυνατότητες ανάπτυξης κώδικα οι οποίες είναι οι παρακάτω:

- **Ασύγχρονος προγραμματισμός**: Το Node.js είναι κατασκευασμένο με βάση ένα μοντέλο εισόδου/εξόδου I/O, που καθοδηγείται από γεγονότα και δεν μπλοκάρει (non-blocking) κατά την εκτέλεση ενός μόνο τμήματος κώδικα αναμένοντας την εκτέλεση του. Αυτή η προσέγγιση, επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών εργασιών χωρίς να αναμένει να ολοκληρωθεί κάθε εργασία πριν προχωρήσει στην επόμενη. Συνεπώς, είναι κατάλληλη επιλογή, για το χειρισμό λειτουργιών που συνδέονται με εισόδους/εξόδους, όπως η ανάγνωση αρχείων, τα ερωτήματα σε βάσεις δεδομένων ή τα αιτήματα δικτύου, τα οποία ενδέχεται να διαρκέσουν αρκετό χρόνο για να ολοκληρωθούν. Αυτό καθιστά το Node.js ιδιαίτερα κατάλληλο για εφαρμογές που απαιτούν υψηλή κλιμάκωση και απόδοση, όπως εφαρμογές πραγματικού χρόνου, APIs και microservices.
- **Μεγάλο οικοσύστημα και κοινότητα**: Το Node.js διαθέτει μια τεράστια και ενεργή κοινότητα προγραμματιστών που συμβάλλουν στην ανάπτυξη και τη βελτίωσή του. Ο Node Package Manager (NPM) παρέχει ένα τεράστιο αριθμό από αποθετήρια (repositories) ανοικτού κώδικα, διευκολύνοντας τους προγραμματιστές να βρίσκουν, να χρησιμοποιούν βιβλιοθήκες για να επιταχύνουν τη διαδικασία ανάπτυξής των project τους.

- **Συμβατότητα πολλαπλών πλατφορμών (Cross-platform):** Το Node.js μπορεί να τρέξει σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των Windows, macOS και Linux, παρέχοντας ευελιξία κατά την ανάπτυξη εφαρμογών.
- **Υποστήριξη σύγχρονων χαρακτηριστικών της JavaScript**: Το Node.js βασίζεται στη μηχανή V8 JavaScript, η οποία ενημερώνεται συνεχώς για την υποστήριξη των πιο πρόσφατων χαρακτηριστικών της ECMAScript. Αυτό επιτρέπει στους προγραμματιστές να αξιοποιούν τις τελευταίες βελτιώσεις και τα νέα χαρακτηριστικά της γλώσσας JavaScript.
- **Σταθερότητα & Ασφάλεια**: Το Node.js συντηρείται και αναπτύσσεται ενεργά, με τακτικές εκδόσεις και εκδόσεις μακροχρόνιας υποστήριξης (Long Term Support - LTS) που εξασφαλίζουν σταθερότητα και ασφάλεια στους χρήστες.

3.4. RabbitMQ

Το RabbitMQ είναι ένας διαμεσολαβητής μηνυμάτων (message broker) (γνωστός επίσης ως ουρά μηνυμάτων ή σύστημα ανταλλαγής μηνυμάτων) ανοικτού κώδικα, που διευκολύνει την επικοινωνία μεταξύ κατανεμημένων συστημάτων υλοποιώντας το Advanced Message Queuing Protocol (AMQP). Το RabbitMQ αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού Erlang και είναι ένας αξιόπιστος, κλιμακούμενος και αποδοτικός message broker, κατάλληλος για το χειρισμό σύνθετων σεναρίων ανταλλαγής μηνυμάτων σε διάφορους κλάδους και εφαρμογές [36]. Το RabbitMQ αναπτύχθηκε αρχικά από την Rabbit Technologies Ltd., η οποία αργότερα εξαγοράστηκε από την VMware το 2010.

Εικόνα 17 Λογότυπο RabbitMQ

Ο διαμεσολαβητής μηνυμάτων RabbitMQ, ενεργεί ως ενδιάμεσος μεταξύ παραγωγών και καταναλωτών, επιτρέποντάς τους, να επικοινωνούν ασύγχρονα χωρίς απευθείας συνδέσεις. Στο πλαίσιο των διαμεσολαβητών μηνυμάτων και της ασύγχρονης επικοινωνίας, οι "παραγωγοί" και οι "καταναλωτές" αναφέρονται στα στοιχεία ή τις εφαρμογές που αλληλεπιδρούν με τον διαμεσολαβητή μηνυμάτων για την αποστολή και τη λήψη μηνυμάτων,

αντίστοιχα. Σε αυτή την αρχιτεκτονική, οι παραγωγοί στέλνουν μηνύματα σε μια ουρά και οι καταναλωτές ανακτούν μηνύματα από την ουρά, επιτρέποντας την ανεξάρτητη κλιμάκωση και ανάπτυξη των δύο τμημάτων. Αυτός ο σχεδιασμός μπορεί να βελτιώσει την ανοχή σε σφάλματα και τη συντήρηση του συστήματος.

Η επιλογή του RabbitMQ έγινε για τους παρακάτω λόγους:

- **Αξιόπιστη παράδοση μηνυμάτων**: Το RabbitMQ παρέχει διάφορους μηχανισμούς για την εξασφάλιση αξιόπιστης παράδοσης μηνυμάτων, όπως η διατήρηση και οι επιβεβαιώσεις μηνυμάτων από το διαμεσολαβητή. Αυτά τα χαρακτηριστικά βοηθούν να διασφαλιστεί ότι τα μηνύματα δεν θα χαθούν, ακόμη και σε περίπτωση αποτυχίας του συστήματος.
- **Επεκτασιμότητα και απόδοση:** Το RabbitMQ έχει σχεδιαστεί για να διαχειρίζεται μεγάλο όγκο μηνυμάτων και μπορεί να κλιμακωθεί οριζόντια με την προσθήκη περισσότερων κόμβων σε μια συστοιχία υπολογιστών (clusters). Αυτό επιτρέπει στο σύστημα να κατανέμει το φορτίο και να αναπτύσσεται ανάλογα με τις ανάγκες, καθιστώντας το κατάλληλο για εφαρμογές υψηλής απόδοσης.
- **Ευέλικτη δρομολόγηση και τύποι ανταλλαγής (routing and exchange types):** Το RabbitMQ υποστηρίζει πολλαπλούς τύπους ανταλλαγής, όπως direct, fanout, topic, and headers, οι οποίοι επιτρέπουν την εξελιγμένη δρομολόγηση μηνυμάτων βάσει κανόνων και προτύπων. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να εφαρμόζουν διάφορα μοτίβα ανταλλαγής μηνυμάτων, όπως publish-subscribe, request-reply και work queues.
- **Υποστήριξη πολλαπλών πρωτοκόλλων ανταλλαγής μηνυμάτων**: Το RabbitMQ υποστηρίζει τα πρωτόκολλα ανταλλαγής μηνυμάτων AMQP, MQTT και HTTP. Αυτό το καθιστά κατάλληλο για ένα ευρύ φάσμα περιπτώσεων χρήσης και επιτρέπει την ενσωμάτωση με διαφορετικά συστήματα και εφαρμογές.
- **Ευρεία υποστήριξη γλωσσών**: Το RabbitMQ παρέχει βιβλιοθήκες-πελάτες για πολλές γλώσσες προγραμματισμού, καθιστώντας εύκολη την ενσωμάτωση με διαφορετικές εφαρμογές και συστήματα.
- **Ασφάλεια και έλεγχος πρόσβασης**: Το RabbitMQ προσφέρει ενσωματωμένη υποστήριξη για κρυπτογράφηση SSL/TLS και παρέχει λεπτομερή έλεγχο πρόσβασης

μέσω των μηχανισμών ελέγχου ταυτότητας και εξουσιοδότησης, εξασφαλίζοντας ασφαλή επικοινωνία μεταξύ των στοιχείων.

- **Ενεργή κοινότητα και υποστήριξη**: Το RabbitMQ διαθέτει ενεργή κοινότητα που συμβάλλει στην ανάπτυξη και τη βελτίωσή του. Επιπλέον, υποστηρίζεται από την VMware, η οποία παρέχει εμπορική υποστήριξη και υπηρεσίες.

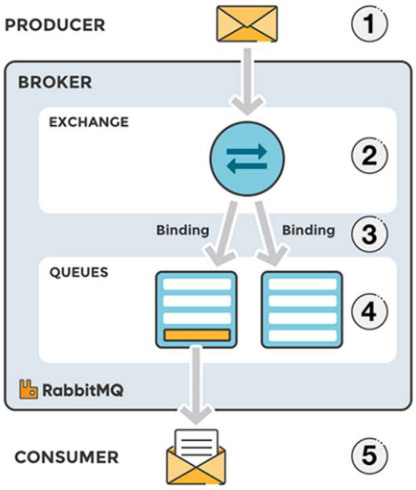
3.4.1. Πως λειτουργεί το RabbitMQ

Στη συγκεκριμένη ενότητα θα αναφέρουμε τον τρόπο λειτουργίας του RabbitMQ.

Εγκατάσταση και ρύθμιση: Αρχικά, πρέπει να γίνει εγκατάσταση του RabbitMQ σε έναν διακομιστή ή σε ένα τοπικό υπολογιστή. Μετά την εγκατάσταση, μπορείτε να εκκινήσετε τον διακομιστή RabbitMQ, ο οποίος θα εκτελείται ως υπηρεσία παρασκηνίου, περιμένοντας συνδέσεις από πελάτες (παραγωγούς και καταναλωτές).

Δημιουργία συνδέσεων: Τόσο οι παραγωγοί όσο και οι καταναλωτές πρέπει να δημιουργήσουν συνδέσεις με τον διακομιστή RabbitMQ χρησιμοποιώντας την κατάλληλη βιβλιοθήκη στη γλώσσα προγραμματισμού που επιθυμούν. Αυτές οι βιβλιοθήκες χειρίζονται τα πρωτόκολλα επικοινωνίας (π.χ. AMQP, MQTT) και παρέχουν ένα εύχρηστο API για την αλληλεπίδραση με το RabbitMQ.

Exchanges and queues: Στο RabbitMQ, τα μηνύματα δεν αποστέλλονται απευθείας σε ουρές. Αντιθέτως, αποστέλλονται στους exchanges, οι οποίες είναι υπεύθυνοι για τη δρομολόγηση των μηνυμάτων σε μία ή περισσότερες ουρές με βάση συγκεκριμένους κανόνες δρομολόγησης. Το RabbitMQ υποστηρίζει διάφορους τύπους exchange (π.χ. direct, fanout, topic, headers) που καθορίζουν τον τρόπο δρομολόγησης των μηνυμάτων στις ουρές. Στις ουρές αποθηκεύονται τα μηνύματα μέχρι να καταναλωθούν [37].



Εικόνα 18 Ροή μηνυμάτων από τον Παραγωγό στον Καταναλωτή

Παραγωγή μηνυμάτων: Όταν ένας παραγωγός θέλει να στείλει ένα μήνυμα, δημιουργεί ένα νεό μήνυμα που περιέχει τα απαραίτητα δεδομένα μαζί με τυχόν πρόσθετες ιδιότητες (π.χ. επικεφαλίδες, προτεραιότητα και χρονική διάρκεια). Στη συνέχεια, ο παραγωγός δημοσιεύει το μήνυμα σε ένα συγκεκριμένο exchange, μαζί με ένα κλειδί δρομολόγησης (routing key). Το routing key χρησιμοποιείται από τον exchange για να καθορίσει σε ποιες ουρές πρέπει να σταλεί το μήνυμα.

Δρομολόγηση μηνυμάτων: Μόλις ένα μήνυμα φτάσει σε ένα exchange, ο exchange δρομολογεί το μήνυμα στις κατάλληλες ουρές (queues), ανάλογα με τον τύπο και τη διαμόρφωσή του. Για παράδειγμα, ένας direct exchange δρομολογεί τα μηνύματα στις ουρές με το αντίστοιχο κλειδί δρομολόγησης, ενώ ένας fanout exchange δρομολογεί τα μηνύματα σε όλες τις δεσμευμένες ουρές, ανεξάρτητα από το κλειδί δρομολόγησης.

Κατανάλωση μηνυμάτων: Οι καταναλωτές εγγράφονται σε μία ή περισσότερες ουρές για να λαμβάνουν μηνύματα. Όταν ένα μήνυμα φτάνει σε μια ουρά, παραδίδεται σε έναν από τους καταναλωτές που έχουν εγγραφεί σε αυτή την ουρά. Ο καταναλωτής επεξεργάζεται το μήνυμα, το οποίο μπορεί να περιλαμβάνει την εκτέλεση μιας εργασίας ή την ενημέρωση της κατάστασης του συστήματος.

Επιβεβαίωση μηνύματος: Μετά την επεξεργασία ενός μηνύματος, ο καταναλωτής στέλνει μια επιβεβαίωση πίσω στο RabbitMQ, υποδεικνύοντας ότι το μήνυμα έχει ληφθεί επιτυχώς και μπορεί να αφαιρεθεί με ασφάλεια από την ουρά. Αυτό διασφαλίζει την αξιόπιστη παράδοση μηνυμάτων και αποτρέπει την απώλεια μηνυμάτων, εάν ένας καταναλωτής αποτύχει να λάβει το μήνυμα.

3.4.2. Εγκατάσταση RabbitMQ σε Ubuntu 18.04

Σε αυτό το σημείο, θα μάθουμε πώς μπορούμε να εγκαταστήσουμε και να διαμορφώσουμε το RabbitMQ στο Ubuntu 18.04. Η εγκατάσταση είναι αρκετά απλή και προϋποθέτει ότι θα εκτελέσουμε την εγκατάσταση με λογαριασμό root, διαφορετικά θα χρειαστεί να προσθέσετε το 'sudo' στις εντολές για να αποκτήσουμε δικαιώματα root. Στη συνέχεια, θα δούμε βήμα προς βήμα την εγκατάσταση του RabbitMQ σε Ubuntu 18.04.

Βήμα 1. Αρχικά, βεβαιωθείτε ότι όλα τα πακέτα του συστήματός σας είναι ενημερωμένα, εκτελώντας τις ακόλουθες εντολές apt-get στο τερματικό.

```
sudo apt-get update
sudo apt-get upgrade
```

Εικόνα 19 Ενημέρωση λειτουργικού συστήματος

Βήμα 2. Θα συνεχίσουμε με την εγκατάσταση της γλώσσας προγραμματισμού Erlang, χρησιμοποιώντας την εντολή:

```
wget http://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc
sudo apt-key add erlang_solutions.asc
sudo apt-get update
sudo apt-get install erlang
sudo apt-get install erlang-nox
```

Εικόνα 20 Εγκατάσταση γλώσσας προγραμματισμού Erlag

Βήμα 3. Εγκατάσταση του RabbitMQ

Πρώτα, θα ενεργοποιήσουμε το RabbitMQ application repository ενώ στη συνέχεια θα προσθέσουμε το δημόσιο κλειδί του RabbitMQ στη λίστα αξιόπιστων κλειδιών για να αποφύγουμε τυχόν προειδοποιήσεις για μη υπογεγραμμένα πακέτα. Έπειτα, θα πρέπει να ενημερώσουμε το σύστημα και να εγκαταστήσουμε το rabbitmq-server.

```
echo "deb http://www.rabbitmq.com/debian/ testing main" >>
/etc/apt/sources.list

wget https://www.rabbitmq.com/rabbitmq-signing-key-public.asc
sudo apt-key add rabbitmq-signing-key-public.asc

sudo apt-get update
sudo apt-get install rabbitmq-server
```

Εικόνα 21 Εγκατάσταση RabbitMQ

Μετά την ολοκλήρωση της εγκατάστασης του RabbitMQ, θα πρέπει να δώσουμε τις κατάλληλες εντολές στο τερματικό για να ξεκινήσουμε, τερματίσουμε αλλά και να προσθέσουμε το RabbitMQ στη λίστα με τις υπηρεσίες που ξεκινούν με την εκκίνηση του λειτουργικού συστήματος.

```
# To automatic enable boot service:
systemctl enable rabbitmq-server

# To start the service:
systemctl start rabbitmq-server

# To stop the service:
systemctl stop rabbitmq-server

# To restart the service:
systemctl restart rabbitmq-server

# To check the status:
systemctl status rabbitmq-server
```

Εικόνα 22 Εκκίνηση RabbitMQ

Βήμα 4. Δημιουργία χρήστη διαχειριστή στο RabbitMQ

Από προεπιλογή, το RabbitMQ δημιουργεί έναν χρήστη με όνομα "guest" και κωδικό πρόσβασης "guest". Μπορείτε επίσης να δημιουργήσετε το δικό σας λογαριασμό διαχειριστή στο RabbitMQ χρησιμοποιώντας τις ακόλουθες εντολές. Αλλάξτε τον κωδικό πρόσβασης μετά τη δημιουργία του νέου χρήστη με έναν ισχυρό κωδικό.

```
sudo rabbitmqctl add_user admin password
sudo rabbitmqctl set_user_tags admin administrator
sudo rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"
```

Εικόνα 23 Δημιουργία χρήστη διαχειριστή στο RabbitMQ

Βήμα 5. Ενεργοποίηση διαχείρισης του RabbitMQ μέσω web

Το RabbitMQ παρέχει κονσόλα διαχείρισης μέσω web για τη διαχείριση του RabbitMQ μέσω γραφικής διεπαφής (GUI). Για να ενεργοποιήσετε την κονσόλα διαχείρισης ιστού εκτελέστε την ακόλουθη εντολή. Η διαδικτυακή κονσόλα διαχείρισης σας βοηθάει στη διαχείριση του διακομιστή RabbitMQ.

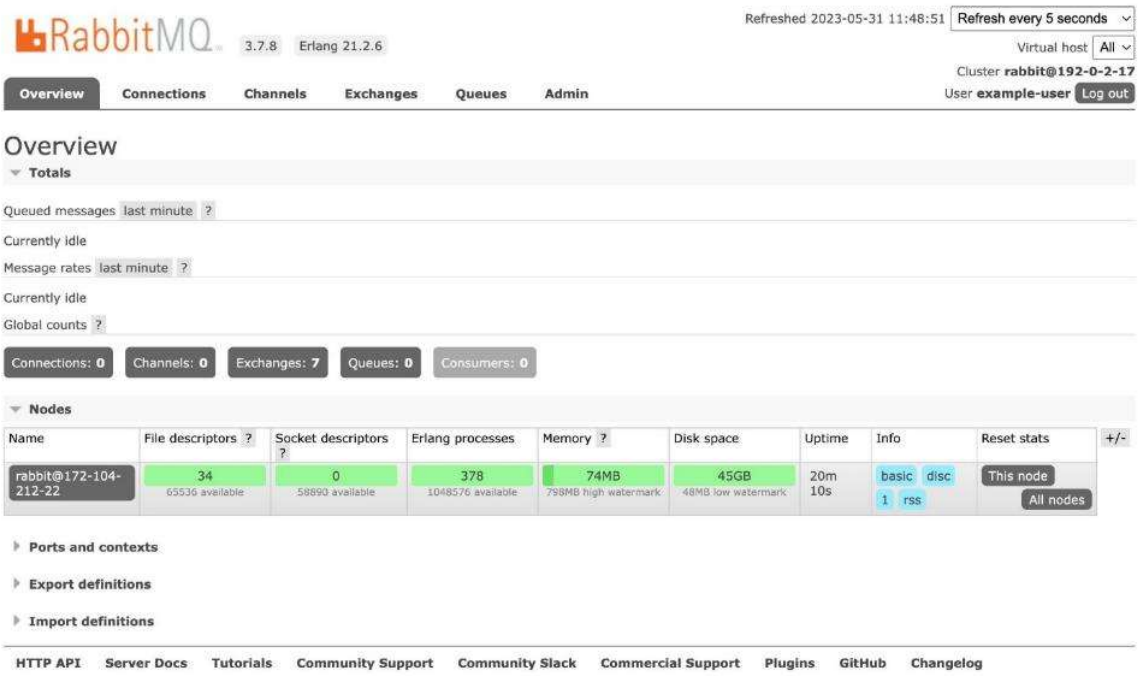
```
sudo rabbitmq-plugins enable rabbitmq_management
```

Εικόνα 24 Ενεργοποίηση διαχείρισης του RabbitMQ μέσω web

Το RabbitMQ θα είναι προσπελάσιμο από θύρα HTTP 15672 (προεπιλεγμένη θύρα). Ανοίξτε το αγαπημένο σας πρόγραμμα περιήγησης και μεταβείτε στη διεύθυνση http://your-domain.com:15672 ή http://server-ip:15672 και θα φορτώσει η παρακάτω σελίδα. Εισάγοντας τα στοιχεία σύνδεσης που δημιουργήσαμε σε προηγούμενο βήμα, θα εισέλθουμε στο διαχειριστικό τμήμα του RabbitMQ.

Εικόνα 25 Οθόνη σύνδεσης RabbitMQ

Στην επόμενη φωτογραφία, απεικονίζεται το διαχειριστικό τμήμα του RabbitMQ.



Εικόνα 26 Διαχειριστικό τμήμα του RabbitMQ

3.5. Thingsboard

3.5.1. Εισαγωγή

Το ThingsBoard είναι μια ανοικτού κώδικα (open source) πλατφόρμα IoT (Internet of Things) που παρέχει λειτουργίες συλλογής, αποθήκευσης, ανάλυσης και οπτικοποίησης δεδομένων από συσκευές IoT. Η πλατφόρμα ThingsBoard είναι σχεδιασμένη για να επιτρέπει την επεκτασιμότητα και την προσαρμοστικότητα σε πολλούς τομείς εφαρμογών IoT, όπως έξυπνες πόλεις, βιομηχανική αυτοματισμός [38].

Εικόνα 27 Λογότυπο Thingsboard

Με το ThingsBoard, οι χρήστες μπορούν να συνδέσουν συσκευές IoT από ποικίλους κατασκευαστές, να διαχειριστούν τα ανεπεξέργαστα δεδομένα (raw data) των αισθητήρων και να παρέχουν πρόσβαση στα συγκεκριμένα δεδομένα στους χρήστες, μέσω ενός ευέλικτου διαδραστικού περιβάλλοντος. Η πλατφόρμα υποστηρίζει μια πληθώρα πρωτοκόλλων συσκευών, όπως MQTT, CoAP, HTTP κτλ, ενώ παρέχει επίσης πλούσιες δυνατότητες διαχείρισης δεδομένων, όπως αποθήκευση, ανάλυση, οπτικοποίηση δεδομένων καθώς επίσης και την αποστολή ειδοποιήσεων, με βάση τα κριτήρια που έχει θέσει ο χρήστης.

3.5.2. Τα χαρακτηριστικά του Thingsboard

Το Thingsboard προσφέρει χαρακτηριστικά, όπως:

Συλλογή δεδομένων: Το Thingsboard μπορεί να συλλέγει δεδομένα από διάφορες συσκευές IoT χρησιμοποιώντας δημοφιλή πρωτόκολλα IoT, όπως MQTT, CoAP και HTTP. Υποστηρίζει ευέλικτα μοντέλα δεδομένων και μπορεί να επεξεργάζεται και να αποθηκεύει δεδομένα είτε πριν από επεξεργασία (raw data) είτε έπειτα από κάποιο μετασχηματισμό.

Επεξεργασία δεδομένων: Το Thingsboard περιλαμβάνει μια ενσωματωμένη μηχανή κανόνων που επιτρέπει στους προγραμματιστές να ορίζουν κανόνες για την επεξεργασία των εισερχόμενων δεδομένων. Αυτή η μηχανή κανόνων μπορεί να χρησιμοποιηθεί για φιλτράρισμα, και μετασχηματισμό των δεδομένων.

Οπτικοποίηση δεδομένων: Το Thingsboard παρέχει τη δυνατότητα δημιουργίας εξατομικευμένων ταμπλό (dashboards), επιτρέποντας στους χρήστες να ενημερώνονται τόσο σε πραγματικό όσο και σε παρελθοντικό χρόνο, για τις μετρήσεις που έχουν συλλέξει οι αισθητήρες.

Ασφάλεια: Το Thingsboard παρέχει ένα ισχυρό μοντέλο ασφάλειας, συμπεριλαμβανομένης της υποστήριξης κρυπτογράφησης SSL/TLS, πιστοποίηση ταυτότητας χρήστη με βάση τους ρόλους και τα δικαιώματα που έχουν ανατεθεί στους χρήστες.

3.5.3. Εγκατάσταση Thingsboard σε Ubuntu 18.04

Σε αυτό το σημείο, θα μάθουμε πώς μπορούμε να εγκαταστήσουμε το Thingsboard στο Ubuntu 18.04. Η εγκατάσταση είναι αρκετά απλή και προϋποθέτει ότι θα εκτελέσουμε την εγκατάσταση με λογαριασμό root, διαφορετικά θα χρειαστεί να προσθέσετε το 'sudo' στις εντολές για να αποκτήσετε δικαιώματα root. Επιπλέον, απαιτείται η εγκατάσταση της βάσης δεδομένων PostgreSQL, στην οποία αποθηκεύει το Thingsboard όλα τα δεδομένα. Για την εκτέλεση του ThingsBoard και της PostgreSQL σε ένα μόνο μηχάνημα θα χρειαστούμε τουλάχιστον 1GB RAM μνήμης. Στη συνέχεια, θα δούμε βήμα προς βήμα την εγκατάσταση του Thingsboard σε Ubuntu 18.04.

Βήμα 1. Εγκατάσταση της Java 11 (OpenJDK): Αρχικά, βεβαιωθείτε ότι όλα τα πακέτα του συστήματός σας είναι ενημερωμένα και στη συνέχεια θα πρέπει να εγκαταστήσουμε την έκδοση 11 της Java, δίνοντας τις παρακάτω εντολές στο τερματικό:

```
sudo apt update
sudo apt install openjdk-11-jdk
```

Εικόνα 28 Εγκατάσταση της Java 11

Σημαντικό: Θα πρέπει να ρυθμίσουμε το λειτουργικό σύστημα (Ubuntu 18.04), έτσι ώστε να χρησιμοποιεί το OpenJDK 11 από προεπιλογή. Για να συμβεί αυτό, θα χρησιμοποιήσουμε την ακόλουθη εντολή:

```
sudo update-alternatives --config java
```

Εικόνα 29 Εκκίνηση OpenJDK 11 στην έναρξη του λειτουργικού

Βήμα 2. Εγκατάσταση του ThingsBoard: Με τις παρακάτω εντολές θα κατεβάσουμε και θα εκτελέσουμε την υπηρεσία του ThingsBoard.

```
sudo wget https://github.com/thingsboard/thingsboard/releases/down-
load/v3.5.1/thingsboard-3.5.1.deb

sudo dpkg -i thingsboard-3.5.1.deb
```

Εικόνα 30 Εγκατάσταση του ThingsBoard

Βήμα 3. Εγκατάσταση & Διαμόρφωση της βάσης δεδομένων PostgreSQL: Με τις παρακάτω οδηγίες θα εγκαταστήσουμε και θα δημιουργήσουμε τη βάση δεδομένων που θα αποθηκεύει το Thingsboard όλα τα δεδομένα.

```
# install **wget** if not already installed:
sudo apt install -y wget

# import the repository signing key:
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo
apt-key add -

# add repository contents to your system:
RELEASE=$(lsb_release -cs)
echo "deb http://apt.postgresql.org/pub/repos/apt/ ${RELEASE}"-pgdg main |
sudo tee  /etc/apt/sources.list.d/pgdg.list

# install and launch the postgresql service:
sudo apt update
sudo apt -y install postgresql-12
sudo service postgresql start
```

Εικόνα 31 Εγκατάσταση της βάσης δεδομένων PostgreSQL

Μόλις εγκατασταθεί η PostgreSQL, θα πρέπει να δημιουργήσετε έναν νέο χρήστη και να ορίσετε τον κωδικό πρόσβασης για το συγκεκριμένο χρήστη. Με τις παρακάτω εντολές

μπορείτε να ορίσετε τον κωδικό πρόσβασης για τον χρήστη της postgresql που επιθυμείτε να αλλάξετε τον κωδικό πρόσβασης.

```
sudo su - postgres
psql
\password
\q
```

Εικόνα 32 Ορισμός κωδικού χρήστη για το χρήστη postgresql

Στη συνέχεια, πατήστε «Ctrl+D» για να επιστρέψετε στην κύρια κονσόλα και να συνδεθείτε στη βάση δεδομένων για να δημιουργήσετε τη βάση δεδομένων που θα επικοινωνεί με το Thingsboard. Η βάση θα ονομάζεται 'thingsboard' και θα δημιουργηθεί με τις παρακάτω εντολές:

```
psql -U postgres -d postgres -h 127.0.0.1 -W
CREATE DATABASE thingsboard;
\q
```

Εικόνα 33 Δημιουργία βάσης 'thingsboard'

Βήμα 4. Διαμόρφωση του configuration file: Για να επεξεργαστούμε το thingsboard.conf θα πρέπει να δώσουμε την παρακάτω εντολή.

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Εικόνα 34 Διαμόρφωση του configuration file

Έπειτα, θα πρέπει να προσθέσουμε τις ακόλουθες γραμμές στο αρχείο ρυθμίσεων. Δε πρέπει να ξεχάσουμε να αντικαταστήσουμε το «PUT_YOUR_POSTGRESQL_PASSWORD_HERE» με τον κωδικό πρόσβασης που δημιουργήσαμε σε προηγούμενο βήμα:

```
# DB Configuration
export DATABASE_TS_TYPE=sql
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=PUT_YOUR_POSTGRESQL_PASSWORD_HERE
# Specify partitioning size for timestamp key-value storage. Allowed values:
DAYS, MONTHS, YEARS, INDEFINITE.
export SQL_POSTGRES_TS_KV_PARTITIONING=MONTHS
```

Εικόνα 35 Διαμόρφωση αρχείο ρυθμίσεων

Βήμα 5: Εκκίνηση της υπηρεσίας ThingsBoard: Για την εκκίνηση της υπηρεσίας θα πρέπει να εκτελέσετε την παρακάτω εντολή.

```
sudo service thingsboard start
```

Εικόνα 36 Εκκίνηση της υπηρεσίας ThingsBoard

Μόλις ξεκινήσει η υπηρεσία, θα πρέπει να μεταβούμε σε ένα browser και να μεταβούμε στον ακόλουθο σύνδεσμο:

```
http://localhost:8080/
```

Εικόνα 37 Σύνδεσμος εισόδου στο Thingsboard

3.6. Μετρητής ενέργειας “Janitza D21 Energy Meter”

3.6.1. Εισαγωγή

Ο μετρητής Janitza D21 είναι ένας μονοφασικός (single-phase) ψηφιακός μετρητής ισχύος που έχει σχεδιαστεί για χρήση σε εμπορικές και οικιακές εφαρμογές, για τη συλλογή και παρακολούθηση ηλεκτρικών παραμέτρων. Κατασκευάζεται από την Janitza electronics GmbH, μία γερμανική εταιρεία που ειδικεύεται σε λύσεις διαχείρισης ενέργειας. Συμβάλει στη μείωση του ενεργειακού κόστους, στη βελτίωση της ενεργειακής απόδοσης και στη μείωση εξάρτησης από τον άνθρακα.

Εικόνα 38 Μετρητής ενέργειας Janitza D21

3.6.2. Χαρακτηριστικά λειτουργείας

Τα πρότυπα IEC 62053-22 και IEC 62053-23 είναι διεθνή πρότυπα που καθορίζουν τις προδιαγραφές για τους μετρητές ενέργειας. Οι κλάσεις ακρίβειας κυμαίνονται από την κλάση 0,1 έως την κλάση 2,0, με την κλάση 0,1 να έχει την υψηλότερη ακρίβεια και την κλάση 2,0 να έχει τη χαμηλότερη ακρίβεια [39].

Στην περίπτωση του αισθητήρα ενέργειας Janitza D21, οι προδιαγραφές "IEC 62053-22: 0,5S, IEC 62053-23: 2" υποδηλώνουν ότι η συσκευή έχει κλάση ακρίβειας 0,5S για τη μέτρηση της ενεργού ενέργειας και κλάση ακρίβειας 2 για τη μέτρηση της άεργου ενέργειας. Αυτό σημαίνει ότι η συσκευή είναι ικανή να μετρά ενεργό ενέργεια με υψηλό βαθμό ακρίβειας, πράγμα που είναι σημαντικό για την ακριβή τιμολόγηση και παρακολούθηση της κατανάλωσης ενέργειας. Η συσκευή είναι επίσης ικανή να μετράει την άεργο ενέργεια με χαμηλότερο επίπεδο ακρίβειας, το οποίο είναι επαρκές για πολλές βιομηχανικές και εμπορικές εφαρμογές. Η ακρίβεια του αισθητήρα ενέργειας αποτελεί σημαντικό στοιχείο κατά την επιλογή μιας συσκευής για την παρακολούθηση και διαχείριση της ενέργειας και διασφαλίζει ότι τα δεδομένα χρήσης ενέργειας είναι αξιόπιστα και ακριβή.

Η τάση λειτουργίας του μετρητή είναι 230V, η οποία είναι η τάση στην οποία έχει σχεδιαστεί να λειτουργεί η συσκευή. Το ρεύμα λειτουργίας είναι 10(60)A, που σημαίνει ότι η συσκευή μπορεί να μετρήσει ρεύμα έως 60A, με ακρίβεια βάσης 10A. Το εύρος συχνοτήτων

προσδιορίζεται από 45Hz έως 65Hz, το οποίο αναφέρεται στο εύρος συχνοτήτων που έχει σχεδιαστεί να μετρά η συσκευή. Επιπλέον, η κατανάλωση ισχύος της συσκευής είναι μικρότερη από 2VA, η οποία αναφέρεται στην απαιτούμενη ισχύ που χρειάζεται η συσκευή για να λειτουργήσει. Το ρεύμα εκκίνησης είναι 0,002Ib, το οποίο αναφέρεται στο ελάχιστο ρεύμα που απαιτείται για την εκκίνηση της συσκευής. Ο ενεργειακός παλμός περιγράφεται ότι έχει πλάτος παλμού 80±20% ms, το οποίο αναφέρεται στη διάρκεια του παλμού που παράγεται κατά τη μέτρηση της ενέργειας. Το σφάλμα ρολογιού προσδιορίζεται ως ≤0,5s, το οποίο αναφέρεται στην ακρίβεια του ρολογιού που χρησιμοποιείται για τη μέτρηση του χρόνου.

Η θύρα RS485 υποστηρίζει το πρωτόκολλο Modbus-RTU, το οποίο είναι ένα πρωτόκολλο επικοινωνίας που χρησιμοποιείται για τη μετάδοση δεδομένων μεταξύ ηλεκτρονικών συσκευών. Η συσκευή διαθέτει βαθμό προστασίας IP51 (πίνακας) / IP20 (περίβλημα), ο οποίος αναφέρεται στο επίπεδο προστασίας που διαθέτει η συσκευή έναντι της σκόνης και του νερού.

Το εύρος θερμοκρασίας λειτουργίας ορίζεται ως (-25～55)°C, το οποίο αναφέρεται στο εύρος θερμοκρασιών για τις οποίες έχει σχεδιαστεί να λειτουργεί η συσκευή. Το εύρος σχετικής υγρασίας είναι (5 έως 95%), το οποίο αναφέρεται στο επίπεδο υγρασίας στο οποίο μπορεί να λειτουργήσει η συσκευή χωρίς να δημιουργηθεί υγρασία στο εσωτερικό της.

Οι παραπάνω τεχνικές προδιαγραφές αποτυπώνονται στον παρακάτω πίνακα:

Πίνακας 1 Χαρακτηριστικά λειτουργίας μετρητή ενέργειας Janitza D21

Item	**Parameter**
Accuracy	IEC 62053-22: 0.5S, IEC 62053-23: 2
Rated voltage	230V
Current	10(60)A
Frequency	45Hz～65 Hz
Wiring	1P2W
Operating voltage range	0.8Un～1.2Un
consumption	< 2VA
Starting current	0.002Ib
Energy pulse	Pulse width (80±20%) ms
Clock error	≤0.5s
RS485 port	Modbus-RTU protocol, baud rate up to 9600bps
IP protection	IP51 (panel) / IP20 (housing)
Operating temperature	(-25～55)℃
Storage temperature	(-25～70)℃
Relative humidity	(5 to 95)% (without condensation)

Οι παράμετροι που απεικονίζονται στον παρακάτω πίνακα, αναφέρονται στην ακρίβεια των μετρήσεων που μπορούν να ληφθούν με τη χρήση του μετρητή ενέργειας D21. Οι τιμές ακρίβειας υποδεικνύουν το μέγιστο δυνατό σφάλμα στη μέτρηση κάθε παραμέτρου, εκφρασμένο ως ποσοστό της πραγματικής τιμής.

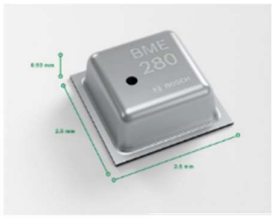
- **Τάση**: Αυτό σημαίνει ότι η τάση που μετρά ο μετρητής ενέργειας μπορεί να αποκλίνει έως και 0,2% από την πραγματική τάση.
- **Ρεύμα**: Η ακρίβεια της μέτρησης ρεύματος καθορίζεται επίσης ως 0,2%, το οποίο σημαίνει ότι το ρεύμα που μετρά ο μετρητής ενέργειας μπορεί να αποκλίνει έως και 0,2% από το πραγματικό ρεύμα.
- **Συχνότητα**: Η ακρίβεια της μέτρησης της συχνότητας καθορίζεται ως ±0,01Hz, το οποίο σημαίνει ότι η συχνότητα που αντλείται από τον μετρητή ενέργειας μπορεί να αποκλίνει έως και 0,01Hz από την πραγματική συχνότητα.
- **Ενεργός ισχύς**: Η ακρίβεια της μέτρησης της ενεργού ισχύος καθορίζεται ως 0,5%, το οποίο σημαίνει ότι η ενεργός ισχύς που αντλείται από τον μετρητή ενέργειας μπορεί να αποκλίνει έως και 0,5% από την πραγματική ενεργό ισχύ.
- **Άεργος ισχύς**: Η ακρίβεια της μέτρησης της άεργου ισχύος ορίζεται επίσης ως 0,5%, το οποίο σημαίνει ότι η άεργος ισχύς που αντλείται από τον μετρητή ενέργειας μπορεί να αποκλίνει έως και 0,5% από την πραγματική άεργο ισχύ.
- **Φαινόμενη ισχύς**: Η ακρίβεια της μέτρησης της φαινόμενης ισχύος καθορίζεται επίσης ως 0,5%, το οποίο σημαίνει ότι η φαινόμενη ισχύς που αντλείται από τον μετρητή ενέργειας μπορεί να αποκλίνει έως και 0,5% από την πραγματική φαινόμενη ισχύ.
- **Συντελεστής ισχύος**: Η ακρίβεια της μέτρησης του συντελεστή ισχύος ορίζεται επίσης ως 0,5%, το οποίο σημαίνει ότι ο συντελεστής ισχύος που μετρά ο μετρητής ενέργειας μπορεί να αποκλίνει έως και 0,5% από τον πραγματικό συντελεστή ισχύος.

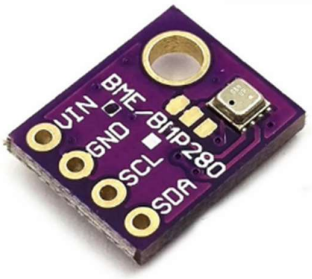
Πίνακας 2 Ακρίβεια μετρήσεων του μετρητή ενέργειας Janitza D21

Function	**Accuracy**
Voltage	0.2
Current	0.2
Frequency	±0.01Hz
Active power	0.5
Reactive power	0.5
Apparent power	0.5
Power factor	0.5

3.7. Αισθητήρας μέτρησης περιβάλλοντος “Bosch BME280 sensor”

3.7.1. Εισαγωγή





Εικόνα 39 Αισθητήρας μέτρησης περιβάλλοντος BOSCH BME280

Ο Bosch BME280 είναι ένας περιβαλλοντικός αισθητήρας που συνδυάζει δυνατότητες ανίχνευσης θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης. Χρησιμοποιείται συνήθως σε διάφορες εφαρμογές, όπως μετεωρολογικοί σταθμοί, συστήματα ελέγχου του κλίματος των εσωτερικών χώρων και συσκευές Internet of Things (IoT). Ο αισθητήρας BME280 παρέχει ακριβείς και αξιόπιστες μετρήσεις, καθιστώντας τον κατάλληλο τόσο για οικιακές όσο και για βιομηχανικές εφαρμογές [40]. Οι μικρές διαστάσεις του και η χαμηλή κατανάλωση ενέργειας επιτρέπουν την εφαρμογή του, σε συσκευές που λειτουργούν με μπαταρία, όπως κινητά τηλέφωνα, μονάδες GPS ή ρολόγια, όπου το μέγεθος και η χαμηλή κατανάλωση ενέργειας αποτελούν βασικές παραμέτρους σχεδιασμού. Η εν λόγω συσκευή συνδυάζει αισθητήρες υψηλής γραμμικότητας και υψηλής ακρίβειας, προσφέροντας μακροχρόνια σταθερότητα και υψηλή ανθεκτικότητα σε ηλεκτρομαγνητική συμβατότητα με ελάχιστη κατανάλωση ενέργειας,

3.7.2. Χαρακτηριστικά λειτουργίας

Ο αισθητήρας είναι συσκευασμένος σε συμπαγή μορφή LGA με μεταλλικό καπάκι και διαστάσεις 2,5 mm x 2,5 mm x 0,93 mm, καθιστώντας τον κατάλληλο για εφαρμογές με περιορισμένο χώρο. Υποστηρίζει τόσο τις ψηφιακές διεπαφές I2C όσο και SPI, επιτρέποντας ευέλικτη και ταχύτατη επικοινωνία με μικροελεγκτές ή άλλες συσκευές. Η κύρια τάση τροφοδοσίας (VDD) κυμαίνεται από 1,71 V έως 3,6 V, ενώ η τάση διασύνδεσης (VDDIO) κυμαίνεται από 1,2 V έως 3,6 V [41].

Όσον αφορά την κατανάλωση ισχύος, ο αισθητήρας παρουσιάζει χαμηλή κατανάλωση ρεύματος, ανάλογα με τον τρόπο λειτουργίας. Σε ρυθμό δειγματοληψίας 1 Hz, καταναλώνει 1,8 μA για μέτρηση υγρασίας και θερμοκρασίας, 2,8 μA για μέτρηση πίεσης και θερμοκρασίας και 3,6 μA για μέτρηση υγρασίας, πίεσης και θερμοκρασίας. Σε κατάσταση αναστολής λειτουργίας, η κατανάλωση ρεύματος μειώνεται σημαντικά στα 0,1 μA, συμβάλλοντας στην εξοικονόμηση ενέργειας.

Το εύρος λειτουργίας του αισθητήρα εκτείνεται από -40°C έως +85°C για τη θερμοκρασία, από 0% έως 100% για την υγρασία και από 300 hPa έως 1100 hPa για την ατμοσφαρική πίεση. Οι αισθητήρες υγρασίας και πίεσης μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν ανεξάρτητα, ανάλογα με τις ανάγκες.

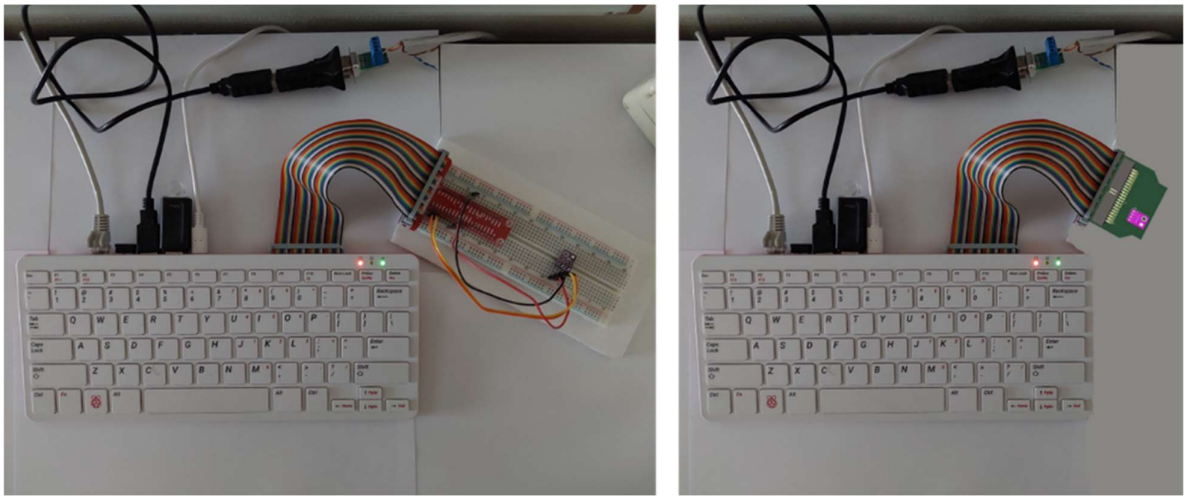
Ο αισθητήρας υγρασίας παρουσιάζει γρήγορο χρόνο απόκρισης 1 δευτερολέπτου , παρέχοντας ταχεία ενημέρωση για τις μεταβολές της υγρασίας. Η ανοχή ακρίβειάς του είναι ±3% σχετική υγρασία και διαθέτει υστέρηση ±1%. Η υστέρηση αναφέρεται στη διαφορά στις ενδείξεις εξόδου του αισθητήρα για μια δεδομένη είσοδο, όταν η τιμή εισόδου αυξάνεται σε σχέση με όταν μειώνεται. Στο πλαίσιο της μέτρησης της υγρασίας του αισθητήρα BME280, μια υστέρηση ±1% σχετικής υγρασίας σημαίνει ότι υπάρχει διαφορά έως και 1% στις ενδείξεις εξόδου του αισθητήρα όταν η υγρασία αυξάνεται σε σύγκριση με όταν μειώνεται. Για να κατανοήσουμε καλύτερα την υστέρηση, ας δούμε ένα παράδειγμα. Ας υποθέσουμε ότι η υγρασία στο περιβάλλον αυξάνεται σταδιακά από 40% σε 60%. Καθώς η υγρασία αυξάνεται, οι ενδείξεις εξόδου του αισθητήρα BME280 θα αυξηθούν ανάλογα. Ωστόσο, όταν η υγρασία αρχίσει να μειώνεται από το 60% στο 40%, οι ενδείξεις εξόδου του αισθητήρα δεν θα μειωθούν αμέσως όπως είχε συμβεί κατά την αύξηση. Αντίθετα, θα υπάρξει μια καθυστέρηση ή διαφορά έως και ±1% σχετικής υγρασίας πριν η έξοδος του αισθητήρα επιστρέψει την ακριβή τιμή υγρασίας.

Για τον αισθητήρα ατμοσφαιρικής πίεσης, το επίπεδο θορύβου RMS είναι 0,2 Pa, που ισοδυναμεί με 1,7 cm, επιτρέποντας ακριβείς μετρήσεις πίεσης. Έχει συντελεστή offset θερμοκρασίας ±1,5 Pa/K, ο οποίος μεταφράζεται σε διακύμανση ±12,6 cm σε αλλαγή θερμοκρασίας κατά 1°C.

3.8. Σχεδίαση πλακέτας PCB σε KiCad

3.8.1. Εισαγωγή

Στόχος της διπλωματικής εργασίας, αποτελεί η σχεδίαση τυπωμένου κυκλώματος Printed Circuit Board – PCB) που θα συνδέεται με το Raspberry Pi 4 και θα συλλέγει τα δεδομένα από τον αισθητήρα περιβάλλοντος Bosch BME280, ώστε η εφαρμογή να προσεγγίζει τη μορφή βιομηχανικού προϊόντος. Για την υλοποίηση της διπλωματικής εργασίας και την άντληση των μετρήσεων από τον αισθητήρα BME280 χρησιμοποιήσαμε το κύκλωμα της παρακάτω αριστερής εικόνας. Στόχος είναι να σχεδιάσουμε με τη βοήθεια του λογισμικού KiCad, το τυπωμένο κύκλωμα της παρακάτω δεξιά εικόνας.



Εικόνα 40 Αισθητήρας BME280 με breadboard και με τυπωμένο κύκλωμα PCB

Η αντικατάσταση ενός breadboard από ένα τυπωμένου κύκλωμα, έχει αρκετά πλεονεκτήματα και είναι τα εξής:

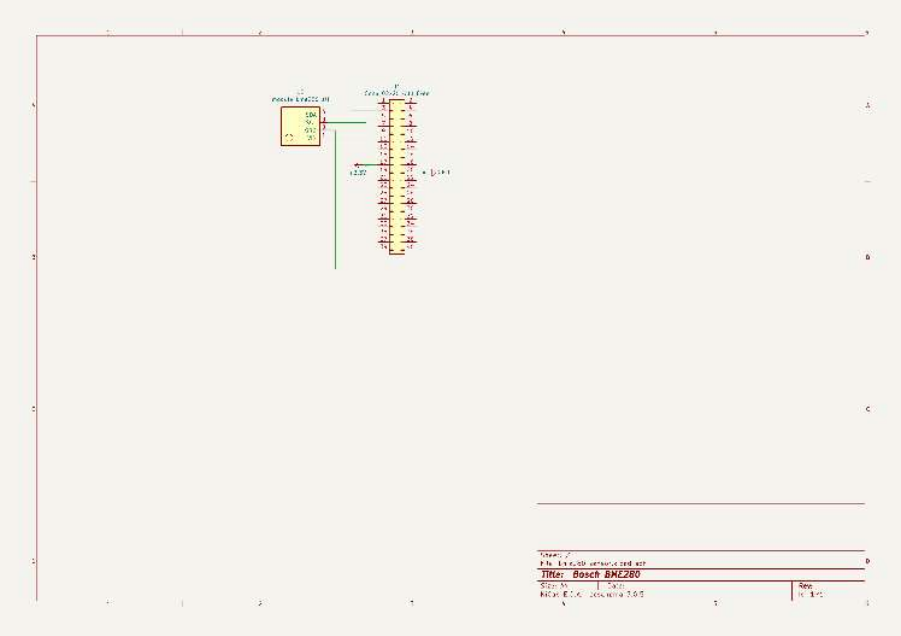
- **Ανθεκτικότητα**: Τα PCB είναι πιο ανθεκτικά και αξιόπιστα από τα breadboards. Τα breadboards είναι ιδανικά για προσωρινές κατασκευές και πρωτότυπα, αλλά δεν προορίζονται για μακροχρόνια χρήση.

- **Ευελιξία**: Τα PCB είναι συνήθως μικρότερα σε μέγεθος και πιο ευέλικτα όσον αφορά την τοποθέτηση των συστατικών εξαρτημάτων. Αυτό είναι ιδιαίτερα σημαντικό σε εφαρμογές όπου ο χώρος είναι περιορισμένος.
- **Σταθερότητα**: Οι συνδέσεις σε ένα PCB είναι πιο σταθερές από τις συνδέσεις σε ένα breadboard. Σε ένα breadboard, οι συνδέσεις μπορεί εύκολα να διακοπούν ή να αποσυνδεθούν, ενώ σε ένα PCB οι συνδέσεις είναι κολλημένες και μόνιμες.
- **Ποιότητα**: Η μεταφορά ενός κυκλώματος από ένα breadboard σε ένα PCB μπορεί να οδηγήσει σε βελτιστοποίηση του κυκλώματος. Τα PCB είναι συνήθως πιο οργανωμένα και δομημένα, ενώ είναι πιο συμπαγή σε σχέση με τα breadboards.

Για να σχεδιάσουμε ένα τυπωμένο κύκλωμα (PCB) στο KiCad για τον αισθητήρα BME280, θα ακολουθήσουμε τα παρακάτω βήματα:

1. **Σχεδίαση Κυκλώματος**: Το συγκεκριμένο βήμα αποτελεί τη δημιουργία του σχηματικού διαγράμματος του κυκλώματος. Στο KiCad, αυτό γίνεται με την επιλογή "Schematic Editor". Σε αυτό το σημείο γίνονται οι συνδέσεις του αισθητήρα BME280 με το Raspberry Pi 4.
2. **Αντιστοίχιση Εξαρτημάτων**: Μετά την ολοκλήρωση του σχηματικού διαγράμματος, θα πρέπει να ορίσουμε αντιστοίχιση μεταξύ του αισθητήρα BME280 με το Raspberry Pi 4, για να προκύψει το τυπωμένο κύκλωμα PCB που θα δούμε παρακάτω.

3.8.2. Διαδικασία σχεδίασης σχηματικού διαγράμματος (Schematic diagram)



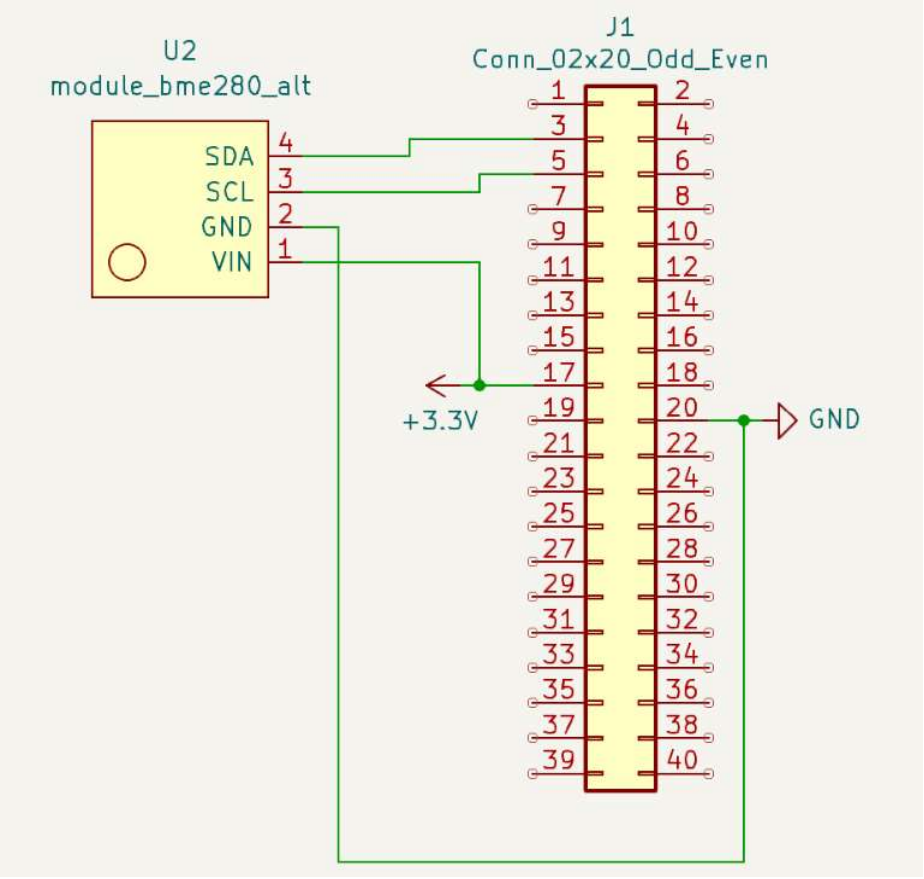
Εικόνα 41 Σχηματικό διάγραμμα (Schematic diagram) στο KiCad

Η δημιουργία του σχηματικού διαγράμματος αποτελεί το βασικό στάδιο της σχεδίασης ενός κυκλώματος. Στο KiCad, η συγκεκριμένη σχεδίαση γίνεται με την επιλογή "Schematic Editor". Η διαδικασία που θα πρέπει να ακολουθήσουμε για τη δημιουργία του σχηματικού είναι:

- Επιλογή και τοποθέτηση των κατάλληλων εξαρτημάτων από τη βιβλιοθήκη συμβόλων του KiCad και εισαγωγή τους στο σχηματικό διάγραμμα.
- Μόλις εισαχθούν τα εξαρτήματα στο σχηματικό διάγραμμα, θα πρέπει να δημιουργηθούν οι κατάλληλες συνδέσεις μεταξύ τους. Αυτό γίνεται με τη χρήση του εργαλείου "Wire" στο KiCad, το οποίο επιτρέπει τη σύνδεση των εξαρτημάτων μεταξύ τους.

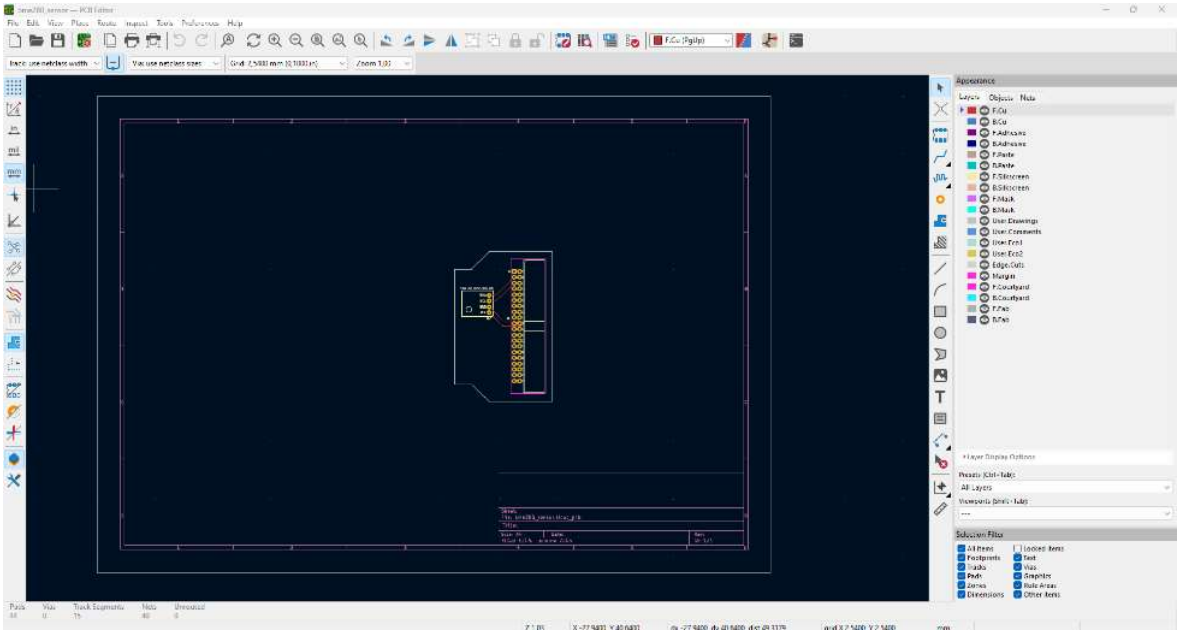
Οι συνδέσεις που πρέπει να γίνουν μεταξύ αισθητήρα BME280 και του ακροδέκτη του Raspberry Pi 4 είναι:

- Οι ακροδέκτες I2C (SDA και SCL) του BME280 θα πρέπει να συνδεθούν στους αντίστοιχους ακροδέκτες I2C (3 και 5) του Raspberry Pi. Ο ακροδέκτης τροφοδοσίας (VIN) του BME280 θα πρέπει να συνδεθεί στον ακροδέκτη 17 με πηγή τροφοδοσίας 3.3V στο Raspberry Pi, ενώ ο ακροδέκτης γείωσης (GND) θα πρέπει να συνδεθεί στον ακροδέκτη 20 που αποτελεί τη γείωση στο Raspberry Pi.
- Αφού επιβεβαιωθεί ότι το σχηματικό διάγραμμα είναι σωστό και πλήρες, μπορούμε να προχωρήσετε στο επόμενο στάδιο της διαδικασίας σχεδίασης που αποτελείται από τη δημιουργία της διάταξης του PCB.



Εικόνα 42 Μεγέθυνση σχηματικού διαγράμματος (Schematic diagram) στο KiCad

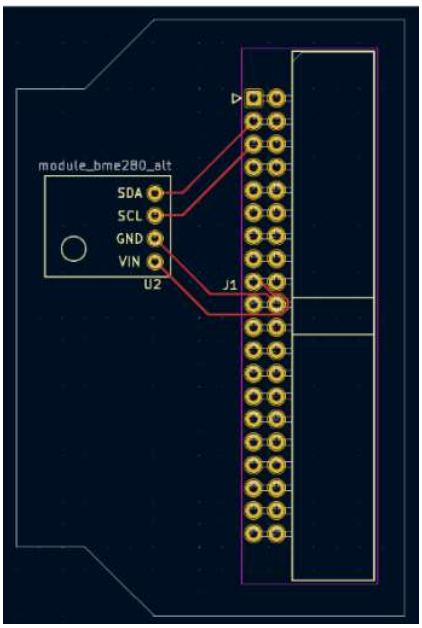
3.8.3. Διαδικασία σχεδίασης τυπωμένου κυκλώματος (PCB)



Εικόνα 43 Σχεδίαση τυπωμένου κυκλώματος (PCB) στο KiCad

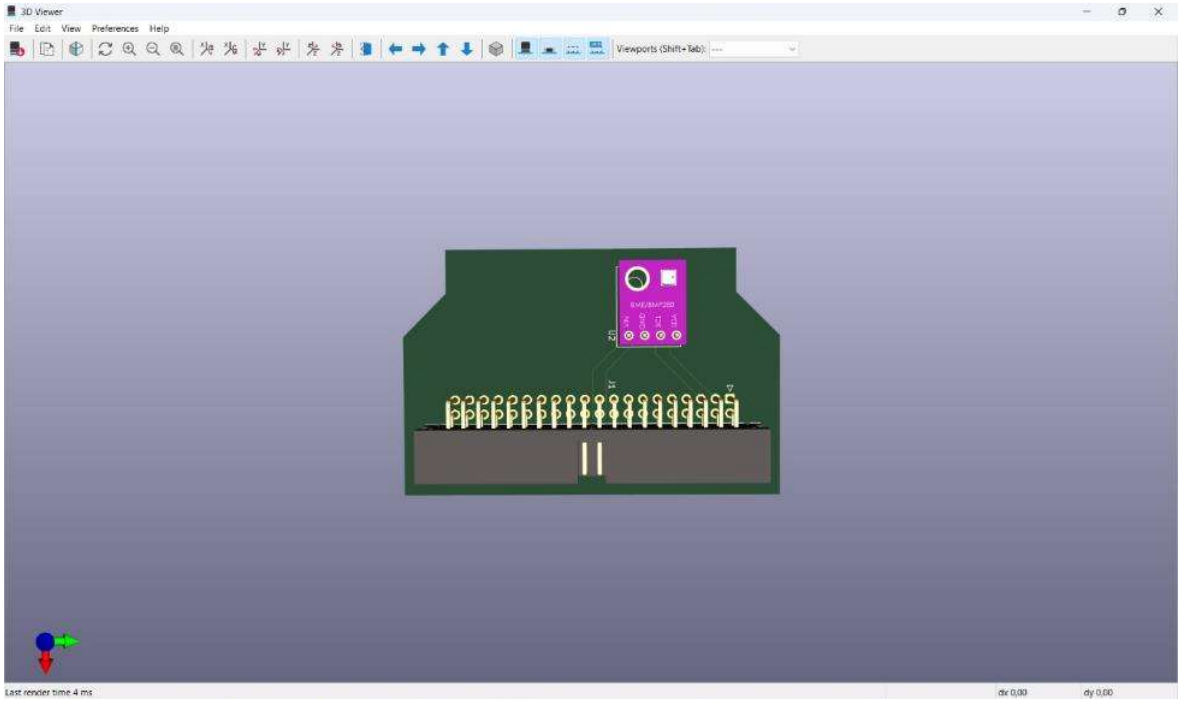
Στη συγκεκριμένη ενότητα τοποθετούμε τα εξαρτήματα του κυκλώματος στην επιφάνεια σχεδίασης και στη συνέχεια πραγματοποιούμε τη δρομολόγηση των συνδέσεων μεταξύ του αισθητήρα περιβάλλοντος BME280 και του ακροδέκτη του Raspberry Pi 4.

Στη διπλανή εικόνα απεικονίζεται το κύκλωμα, μετά τη δρομολόγηση των συνδέσεων. Να τονίσουμε σε αυτό το σημείο, ότι έχουμε προσθέσει τα όρια της πλακέτας, έτσι ώστε να γνωρίζει ο κατασκευαστής, σε ποια σημεία θα πρέπει να κοπεί, όταν τη στείλουμε προς υλοποίηση.



Εικόνα 44 Δρομολόγηση συνδέσεων στο KiCad

Επιπλέον, το KiCad προσφέρει τη δυνατότητα να απεικονίσουμε τη πλακέτα που έχουμε σχεδιάσει και υλοποιήσει, σε 3D μορφή, όπως ακριβώς θα είναι το τελικό προϊόν που θα παραχθεί από το εργοστάσιο, όταν τη στείλουμε προς υλοποίηση.



Εικόνα 45 Προεπισκόπηση πλακέτας σε 3D μορφή

Τέλος, για να στείλουμε τα σχέδια της πλακέτας προς υλοποίηση σε ένα εργοστάσιο της επιλογής μας, θα πρέπει να επιλέξουμε από το KiCad την εξαγωγή των gerber files και των drill files. Τα συγκεκριμένα αρχεία μπορούμε να τα ανεβάσουμε στο website του κατασκευαστή που επιθυμούμε να κατασκευάσει την πλακέτα και θα μας ενημερώσει για το χρόνο υλοποίησης - παράδοσης καθώς και για το συνολικό κόστος.

3.9. Κόστος εξοπλισμού διπλωματικής εργασίας

Για την υλοποίηση της παρούσας διπλωματικής εργασίας και τη δημιουργία της εφαρμογής σε node.js, απαραίτητη ήταν η προμήθεια του παρακάτω εξοπλισμού.

Πίνακας 3 Κόστος εξοπλισμού διπλωματικής εργασίας

Εξοπλισμός	**Τιμή**
Breadboard	8€
Raspberry Pi 400 Personal Computer kit	108€
Μετρητής ηλεκτρικής ενέργειας “Janitza D21 Energy Meter”	80,60€
Μετρητής περιβάλλοντος “Bosch BME280”	19,34€
Καλώδια σύνδεσης (Jumper Wires)	5€
Raspberry Pi 40 Pin GPIO Ribon Cable Extension Board + GPIO Cobbler Extension Board για τη σύνδεση του Raspberry PI 400 με το Breadboard.	3€
USB Stick 3.2 (32GB) to boot Raspberry Pi OS	15€
Σύνολο:	**238.94€**

Κεφάλαιο 4. Εφαρμογή συλλογής μετρήσεων αισθητήρων (Ανάλυση κώδικα)

4.1. Εισαγωγή

Στο συγκεκριμένο κεφάλαιο, αρχικά θα αναλύσουμε τις συνδέσεις που πρέπει να επιτευχθούν, έτσι ώστε να είναι σε θέση, οι μετρητές να επικοινωνήσουν με τον ελεγκτή Raspberry Pi 4. Στη συνέχεια, θα προχωρήσουμε στην ανάλυση του κώδικα με τον οποίο αντλούμε τα δεδομένα από τους αισθητήρες. Τέλος, θα γνωρίσουμε τα γραφήματα που δημιουργήθηκαν στο Thingsboard, για να πληροφορείτε ο χρήστης με γρήγορο και εποπτικό τρόπο για τις τρέχουσες ή προγενέστερες μετρήσεις, που κατέγραψαν οι αισθητήρες.

4.2. Σύνδεση & Ανάλυση του μετρητή ενέργειας “Janitza D21 Energy Meter”

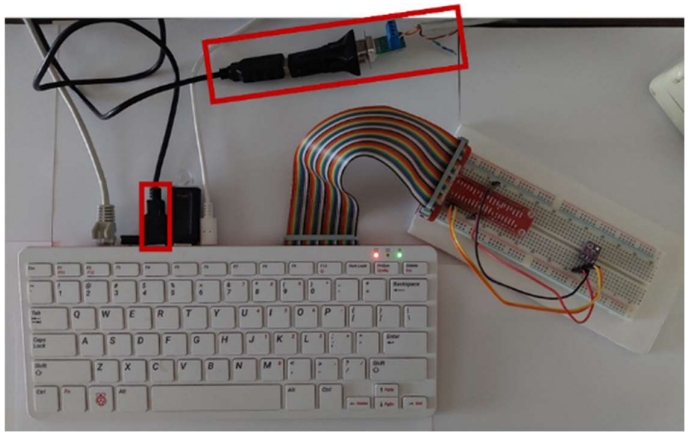
4.2.1. Σύνδεση του “Janitza D21 Energy Meter” στον ηλεκτρολογικό πίνακα

Η σύνδεση του μετρητή ενέργειας “Janitza D21 Energy Meter” με τον ηλεκτρολογικό πίνακα έγινε από αδειούχου ηλεκτρολόγο. Η τοποθέτηση του μετρητή έγινε σε ράγα DIN εντός του ηλεκτρικού πίνακα. Για τη συνδεσμολογία χρησιμοποιήθηκε το user manual του μετρητή, στο οποίο απεικονίζονται επακριβώς οι συνδέσεις. Η είσοδος της φάσης L από τον πίνακα προς το μετρητή επιτυγχάνεται χρησιμοποιώντας τη θύρα 1, ενώ η έξοδος τη θύρα 2. Αντίστοιχα, για τον ουδέτερο N, η είσοδος επιτυγχάνεται χρησιμοποιώντας τη θύρα 3, ενώ η έξοδος τη θύρα 4.

Εικόνα 46 Σύνδεση του Janitza D21 στον ηλεκτρολογικό πίνακα

4.2.2. Σύνδεση του “Janitza D21 Energy Meter” στο Raspberry PI

Για τη σύνδεση του μετρητή (μέσω Modbus πρωτοκόλλου) με το Raspberry Pi και την αποστολή των δεδομένων, απαραίτητος είναι ο μετατροπέας της εταιρείας DIGITUS. Ο μετατροπέας DIGITUS USB σε σειριακό RS485 είναι μια συσκευή που επιτρέπει την επικοινωνία μεταξύ ενός υπολογιστή ή μιας άλλης συσκευής η οποία διαθέτει USB και μιας συσκευής που χρησιμοποιεί επικοινωνία RS485. Μετατρέπει το σήμα USB από τον υπολογιστή σε σήμα RS485 που μπορεί να χρησιμοποιηθεί για την επικοινωνία με τη συσκευή RS485.



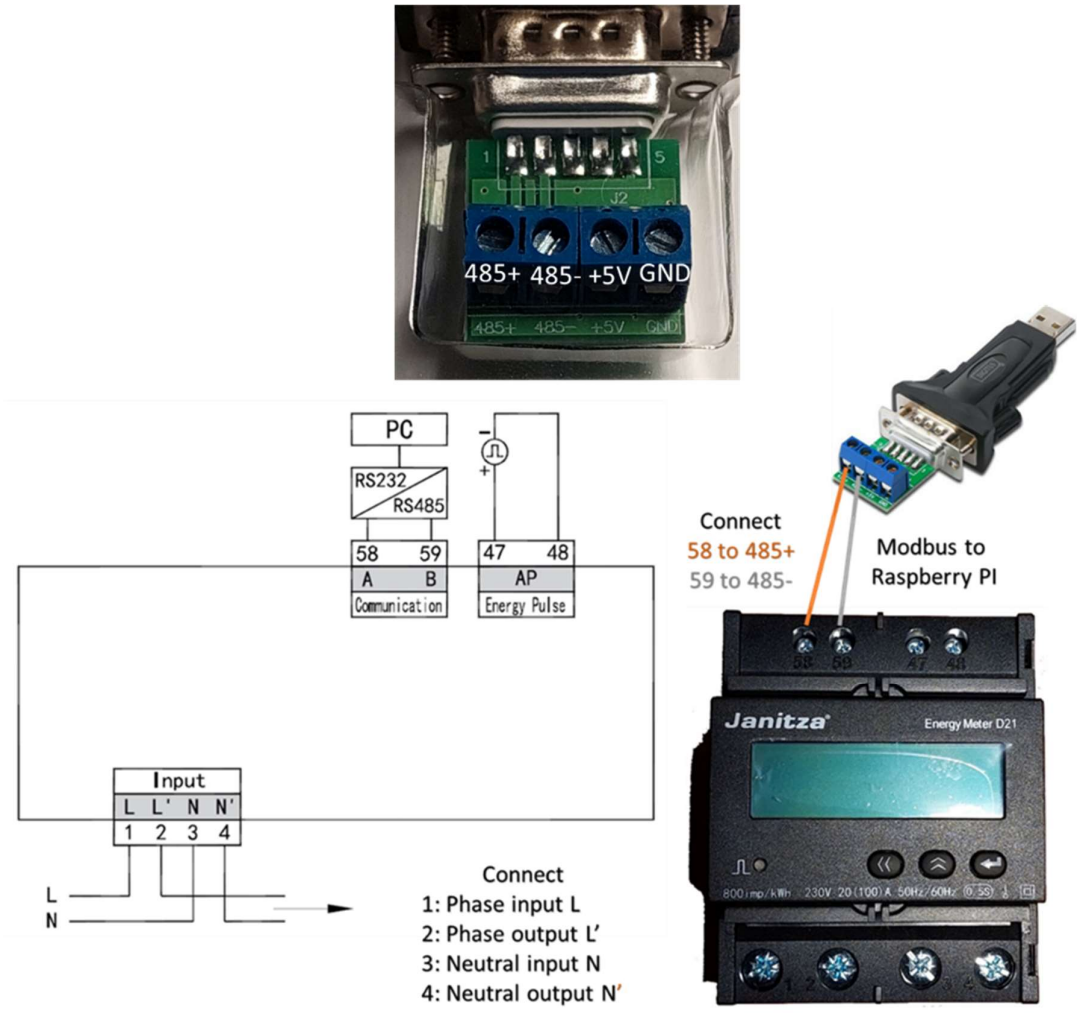
Εικόνα 47 Σύνδεση του Janitza D21 Energy στο Raspberry PI

Εικόνα 48 Μετατροπέας USB to serial RS485

Ο μετατροπέας έχει σχεδιαστεί για να συνδεθεί στη θύρα USB ενός υπολογιστή και παρέχει μια διεπαφή RS485 που μπορεί να συνδεθεί σε μια συσκευή RS485 χρησιμοποιώντας ένα καλώδιο δικτύου (UTP). Ο μετατροπέας τροφοδοτείται από τη θύρα USB του υπολογιστή, οπότε δεν απαιτείται πρόσθετη παροχή ρεύματος. Ο μετατροπέας υποστηρίζει ένα ευρύ φάσμα ρυθμών baud και μορφών δεδομένων, επιτρέποντάς του να επικοινωνεί με μια ποικιλία συσκευών μέσω RS485. Υποστηρίζει επίσης το πρωτόκολλο επικοινωνίας Modbus, το οποίο είναι ένα κοινό πρωτόκολλο που χρησιμοποιείται για την επικοινωνία μέσω RS485.

Για να πραγματοποιηθεί η σύνδεση του μετρητή ενέργειας Janitza D21 με το μετατροπέα DIGITUS, χρησιμοποιήθηκαν οι θύρες 58 και 59 οι οποίες καταλήγουν στον ακροδέκτη 485+ και 485- αντίστοιχα.

Η συνδεσμολογία του μετρητή απεικονίζεται αναλυτικά στο παρακάτω σχήμα:



Εικόνα 49 Σύνδεση μετρητή ενέργειας Janitza D21 με το μετατροπέα DIGITUS

4.2.3. Καταγραφή ενέργειας με τη χρήση του “Janitza D21 Energy Meter”

Για τις ανάγκες τις διπλωματικής εργασίας χρησιμοποιήθηκε ο μετρητής ενέργειας “Janitza D21 Energy Meter”, με τη βοήθεια του οποίου πραγματοποιήθηκε η καταγραφή των παρακάτω ενεργειακών παραμέτρων.

Οι μετρήσεις που καταγράφηκαν είναι:

Πίνακας 4 Καταγραφή μετρήσεων ενέργειας με το Janitza D21

Μέτρηση	**Σύμβολο μέτρησης**	**Μονάδα μέτρησης**
Voltage - Τάση	V	V
Ρεύμα – Current	I	A
Πραγματική ισχύς - Active power	P	kW
Άεργος ισχύς– Reactive power	Q	kVAR
Φαινόμενη ισχύς – Apparent power	S	kVA
Power factor (cos)		
Συχνότητα – Frequency	F	Hz
Ενέργεια που καταναλώθηκε - Active energy	Ea	kWh
Άεργος ενέργεια - Reactive energy	Er	kvarh

Πίνακας 5 Καταγραφή δεδομένων στην οθόνη του Janitza D21

Οθόνη μετρητή 'Janitza D21' user manual	**Περιγραφή / Εξήγηση μέτρησης**	**Οθόνη Μετρητή 'Janitza D21'**
	Voltage - Τάση (V): V = 220.0 V	
	Current – Ρεύμα (A): I = 35.00 A	
	Active power - Ενεργός ισχύς (kW): P = 7.700 kW	
	Reactive power - Άεργος ισχύς (kvar): Q = -0.006 kvar	
	Apparent power - Φαινόμενη ισχύς (S): S = 7.700 kVA	
	Power factor - Συντελεστής ισχύος (cos): PF = 1.000	
	Frequency – Συχνότητα (Hz): F = 50.00 Hz	

	Active energy - Ηλεκτρική ενέργεια ή ενεργός ενέργεια (kWh): EP+ =780.62 kWh	
	Reactive energy - Άεργος ενέργεια (kvarh)： EQ+ = 18.80 kvarh	

4.2.4. Ανάλυση κώδικα για το μετρητή ενέργειας “Janitza D21 Energy Meter”

Για τη διασύνδεση του μετρητή κατανάλωσης ενέργειας με το Raspberry PI 4 και την εξαγωγή των δεδομένων, δημιουργήθηκε ο παρακάτω κώδικας σε node.js. Ολόκληρος ο κώδικας μπορεί να βρεθεί στο παράρτημα, στο τέλος της διπλωματικής εργασίας. Στο παρόν τμήμα, θα αναλύσουμε τα κυριότερα σημεία του κώδικα που είναι υπεύθυνα για την άντληση των δεδομένων.

4.2.4.1. Επικοινωνία Μετρητή Ενέργειας μέσω Modbus RTU και Raspberry Pi 4

Αρχικά, εισάγεται η βιβλιοθήκη "modbus-serial" για την εγκαθίδρυση επικοινωνίας με τον μετρητή ενέργειας χρησιμοποιώντας το πρωτόκολλο Modbus RTU μέσω μιας σειριακής σύνδεσης. Έπειτα, δημιουργείται ένα νέο instance της κλάσης ModbusRTU, με τη βοήθεια του οποίου, παρέχετε πρόσβαση σε διάφορες μεθόδους και συναρτήσεις που εμπεριέχει η βιβλιοθήκη "modbus-serial", όπως είναι η δημιουργία σύνδεσης με μια συσκευή Modbus και η ανάγνωση ή εγγραφή δεδομένων από/προς τους καταχωρητές.

Στη συνέχεια, καλείται η μέθοδος client.connectRTUBuffered() για να δημιουργηθεί μια σειριακή σύνδεση με το μετρητή ενέργειας, καθορίζοντας τη σειριακή θύρα ('/dev/ttyUSB0') και τον ρυθμό μετάδοσης δεδομένων (baud rate) στην τιμή 9600 (προεπιλεγμένη τιμή του ‘Janitza D21’). Η σειριακή θύρα ('/dev/ttyUSB0') αναφέρεται στον μετατροπέα “USB to Serial RS485 Converter” που είναι συνδεδεμένος με το Raspberry Pi 4 και γνωρίσαμε παραπάνω.

Στην επόμενη γραμμή εκτελείται η μέθοδος setID(2) στο αντικείμενο client για να ορίσει το Modbus slave ID (επίσης γνωστό ως αναγνωριστικό μονάδας) για τη σύνδεση με το μετρητή ενέργειας. Το Modbus slave ID (αναγνωριστικό μονάδας) έχουμε τη δυνατότητα να το πληροφορηθούμε, είτε από το εγχειρίδιο χρήσης (manual) του μετρητή, είτε κάνοντας χρήση της οθόνης που διαθέτει ο μετρητής. Στο μετρητή που χρησιμοποιήσαμε, το αναγνωριστικό μονάδας ορίζεται στην τιμή 2.

```
const ModbusRTU = require("modbus-serial");
const client = new ModbusRTU();
client.connectRTUBuffered('/dev/ttyUSB0', { baudRate: 9600 });
client.setID(2);
```

Εικόνα 50 Αρχικοποίηση επικοινωνίας Modbus με Janitza D21

4.2.4.2. Αρχικοποίηση RabbitMQ

Στην αρχή του παρακάτω κώδικα, δημιουργήθηκε το αντικείμενο RABBIT_PARAMS που περιέχει την κατάλληλή παραμετροποίηση για τη σύνδεση και την επικοινωνία με το διαμεσολαβητή μηνυμάτων (message broker) RabbitMQ. Το αντικείμενο RABBIT_PARAMS έχει τις ακόλουθες ιδιότητες:

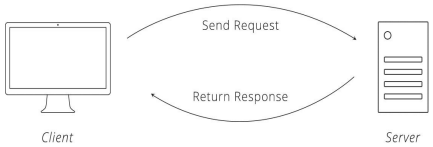
- **HOST**: Το όνομα κεντρικού υπολογιστή του διακομιστή RabbitMQ, το οποίο λαμβάνεται από ένα αντικείμενο config χρησιμοποιώντας την ιδιότητα RABBIT.HOST.
- **PORT**: Ο αριθμός θύρας στην οποία 'ακούει' ο διακομιστής RabbitMQ, ο οποίος λαμβάνεται από το αντικείμενο config χρησιμοποιώντας την ιδιότητα RABBIT.PORT.
- **USER**: Το όνομα χρήστη που πρέπει να χρησιμοποιείται κατά τη σύνδεση με τον διακομιστή RabbitMQ, το οποίο λαμβάνεται από το αντικείμενο config χρησιμοποιώντας την ιδιότητα RABBIT.SMART_USER.
- **PASSWORD**: Ο κωδικός πρόσβασης που πρέπει να χρησιμοποιείται κατά τη σύνδεση με τον διακομιστή RabbitMQ, που λαμβάνεται από το αντικείμενο config χρησιμοποιώντας την ιδιότητα RABBIT.SMART_PASSWORD.
- **ExchangeName**: Το όνομα της ανταλλαγής που θα χρησιμοποιηθεί για τη δημοσίευση και την εγγραφή μηνυμάτων.
- **ExchnageType**: Ο τύπος της ανταλλαγής, ο οποίος στην προκειμένη περίπτωση είναι 'topic'.
- **SmartEnergyMeterRoutingKey**: Το κλειδί δρομολόγησης που χρησιμοποιείται για το φιλτράρισμα των μηνυμάτων κατά την ανταλλαγή.
- **IsDurable**: Λογική (Boolean) τιμή που υποδεικνύει αν η ανταλλαγή θα πρέπει να διατηρηθεί σε περίπτωση επανεκκίνησης του διαμεσολαβητή ή όχι. Όταν μια

ανταλλαγή δηλώνεται ως durable, αυτό σημαίνει ότι το RabbitMQ θα την αποθηκεύσει στο δίσκο, ώστε να μπορεί να ανακτηθεί σε περίπτωση αποτυχίας ή επανεκκίνησης του διαμεσολαβητή.

4.2.4.3. Κλήση μεθόδου 'd21_get' (HTTP GET request)

Επιπρόσθετα, χρησιμοποιώντας το framework Express.js και τη μέθοδο router.get() για τον καθορισμό ενός χειριστή διαδρομής για εισερχόμενα αιτήματα HTTP GET. Με άλλα λόγια, χρησιμοποιώντας τη συγκεκριμένη μέθοδο, καλούμε ένα HTTP GET request είτε με τη βοήθεια του Postman είτε ενός browser, με σκοπό τον έλεγχο ή την αποσφαλμάτωση (debugging) του κώδικα. Τη συγκεκριμένη λειτουργεία θα τη γνωρίσουμε στην επόμενη ενότητα.

Ακολουθεί ο block κώδικας (Try-catch), όπου εμπεριέχεται η κυρίως λειτουργία της εφαρμογής. Χρησιμοποιήθηκε η βιβλιοθήκη 'winstonLogger' η οποία μας βοηθάει στην καταγραφή των logs τόσο κατά τον εντοπισμό σφαλμάτων (erros) winstonLogger.error() όσο και κατά τη διάρκεια της αποσφαλμάτωσης (debug). Στις επόμενες γραμμές, ορίζουμε το όνομα του virtual_host = 'smart_meters' έτσι όπως το έχουμε ορίσει στο διαμεσολαβητή RabbitMQ. Μετά, ακολουθεί ο έλεγχος για το εάν μπορεί να πραγματοποιηθεί σύνδεση με το διαμεσολαβητή, λαμβάνοντας ως ορίσματα το όνομα του παραπάνω virtual_host και των παραμέτρων RABBIT_PARAMS. Εάν η σύνδεση δεν μπορεί να επιτευχθεί, τότε εμφανίζεται το μήνυμα " No Rabbit channel available !" και τερματίζεται η εκτέλεση του κώδικα. Μόλις η σύνδεση με το διαμεσολαβητή επιτευχθεί, καλείται η συνάρτηση GetDataSmartMeterHome, η οποία είναι υπεύθυνη για την άντληση των δεδομένων από τον έξυπνο μετρητή και θα αναλυθεί στη συνέχεια. Η επιστροφή της μεθόδου εκχωρείται στη μεταβλητή measure και στη συνέχεια δρομολογείται το μήνυμα για την αποστολή στο διαμεσολαβητή RabbitMQ χρησιμοποιώντας τη μέθοδο SendToRabbitmq() και τις κατάλληλες παραμέτρους που αναφέραμε προηγουμένως. Τέλος, αποστέλλουμε τα δεδομένα της μεταβλητής measure στον πελάτη – χρήστη, ως απάντηση (response) του HTTP GET request αιτήματος που υπέβαλε προς την εφαρμογή, όπως φαίνεται στην εικόνα.



Εικόνα 51 HTTP request & response to API call

```
const RABBIT_PARAMS = {
    HOST: config.RABBIT.HOST,
    PORT: config.RABBIT.PORT,
    USER: config.RABBIT.SMART_USER,
    PASSWORD: config.RABBIT.SMART_PASSWORD,
    ExchangeName: 'amq.topic',
    ExchnageType: 'topic',
    SmartEnergyMeterRoutingKey: 'meter.master',
    IsDurable: true
}


router.get('/d21_get', async function (req, res, next) {
    try {
        winstonLogger.log('debug', req.query);
        const virtual_host = 'smart_meters';
        const channel = await rabbit.GetChannel(virtual_host, RABBIT_PARAMS);

        if (!channel) {
            res.send("No Rabbit channel available !");
            return;
        }

        let measure = await GetDataSmartMeterHome(client, 0, 22);
        rabbit.SendToRabbitmq(channel, RABBIT_PARAMS.ExchangeName,
RABBIT_PARAMS.SmartEnergyMeterRoutingKey, measure);

        res.send(measure);

    } catch (e) {
        winstonLogger.error(e);
        res.send(e);
    }
});
```

Εικόνα 52 Κλήση της μεθόδου d21_get του μετρητή Janitza D21

4.2.4.4. Η συνάρτηση GetDataSmartMeterHome()

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την επικοινωνία με το μετρητή ενέργειας “Janitza D21 Smart Energy Meter” και την άντληση των ανεπεξέργαστων δεδομένων – μετρήσεων (raw data), έτσι ώστε να επιστραφούν στο κυρίως σώμα της εφαρμογής που είδαμε παραπάνω, με σκοπό την ενημέρωση και την πληροφόρηση του τελικού χρήστη. Για να συμβεί αυτό όμως, θα πρέπει να αντλήσουμε τα δεδομένα από το μετρητή χρησιμοποιώντας κατάλληλες μεθόδους, καθώς και να συμβουλευτούμε το εγχειρίδιο του μετρητή, για να είμαστε σε θέση να κατανοήσουμε ποιους καταχωρητές (registers), θα πρέπει να προσπελάσουμε ούτως ώστε να διαβάσουμε τις καταχωρήσεις τους.

Κατά την κλήση της η συνάρτηση λαμβάνει τρία ορίσματα: το αντικείμενο **client**, το **reg_start**, **reg_finsih**, τα όποια αναφέρονται στη σύνδεση του μετρητή χρησιμοποιώντας το πρωτόκολλο Modbus, τον αριθμό του καταχωρητή (register start) από τον οποίο θα ξεκινήσει η προσπέλαση και τον αριθμό του καταχωρητή (register finish) όπου θα τελειώσει η προσπέλαση, αντίστοιχα.

Στη συνέχεια, υπάρχει ο block κώδικας (Try-catch), όπου χρησιμοποιούμε τη βιβλιοθήκη ‘moment-timezone’ για να μετατρέψουμε την τρέχουσα ημερομηνία και ώρα (timestamp) στη ζώνη ώρας "Europe/Athens" σε χιλιοστά του δευτερολέπτου.

Έπειτα, καλείται η μέθοδος readHoldingRegisters στο αντικείμενο client, με ορίσματα reg_start και reg_finish με σκοπό να επιστραφούν οι μετρήσεις από τις συγκεκριμένες θέσεις μνήμης που δίνουμε σαν ορίσματα. Στη συγκεκριμένη γραμμή, λαμβάνουμε τις μετρήσεις από τις θέσεις μνήμης που έχουμε δώσει σαν ορίσματα και απεικονίζονται στον παρακάτω πίνακα. Με αυτό τον τρόπο αντλούνται οι μετρήσεις και αποθηκεύονται στη μεταβλητή measure.

Πίνακας 6 Καταχωρητές για την άντληση των δεδομένων από το Janitza D21

Address	**Format**	**Data description**	**Unit**	**R/W**
00000-00001	Float	Voltage - Τάση	V	R
00002-00003	Float	Current – Ρεύμα	A	R
00004-00005	Float	Active power – Ενεργός ισχύς	kW	R
00006-00007	Float	Reactive power - Άεργος ισχύς	kvar	R
00008-00009	Float	Apparent power - Φαινόμενη ισχύς	kVA	R
00010-00011	Float	Power factor - Συντελεστής ισχύος		R
00012-00013	Float	Frequency – Συχνότητα	Hz	R
00014-00015	Float	Active energy - Ηλεκτρική ενέργεια ή ενεργός ενέργεια	kWh	R
00018-00019	Float	Reactive energy - Άεργος ενέργεια	kvarh	R

Στη συνέχεια, θα πρέπει να μορφοποιηθούν οι μετρήσεις με κατάλληλο τρόπο έτσι ώστε να μπορούν να αναγνωσθούν από τον χρήστη. Γι' αυτό θα κάνουμε χρήση μεθόδων από τη JavaScript όπως και της κλάσης Buffer της Node.js έτσι ώστε να λάβουμε τις μετρήσεις και να τις εκχωρήσουμε σε κατάλληλη μεταβλητή.

Τα βήματα που θα ακολουθήσουμε για να αναλύσουμε την κάθε ανάλυση είναι:

- measure.buffer: Ο buffer δεδομένων που περιέχει τις μετρήσεις από τον έξυπνο μετρητή. Αναφέρεται σε ένα αντικείμενο buffer, το οποίο είναι μια περιοχή προσωρινής αποθήκευσης δυαδικών δεδομένων.
- readFloatBE(index): Αυτή η μέθοδος διαβάζει έναν αριθμό κινητής υποδιαστολής 32 bit από το buffer στην καθορισμένη θέση index. Το BE στην readFloatBE σημαίνει "Big Endian". Αυτή είναι η σειρά byte με την οποία αποθηκεύεται ο αριθμός κινητής υποδιαστολής στο buffer. Η Big-endian (BE) είναι μία από τις μορφές διάταξης byte που χρησιμοποιούνται στα συστήματα υπολογιστών. Σε ένα σύστημα Big-endian (BE), το πιο σημαντικό byte (MSB), το οποίο περιέχει τα bit υψηλότερης σειράς, αποθηκεύεται στη χαμηλότερη διεύθυνση μνήμης, ενώ το λιγότερο σημαντικό byte (LSB), το οποίο περιέχει τα bit χαμηλότερης σειράς, αποθηκεύεται στην υψηλότερη διεύθυνση μνήμης.
- toFixed(2): Αυτή η μέθοδος καλείται στον αριθμό που επιστρέφεται από την readFloatBE(index). Στρογγυλοποιεί τον αριθμό στην πλησιέστερη τιμή διατηρώντας έως και 2 δεκαδικά ψηφία.
- parseFloat(): Αυτή η συνάρτηση μετατρέπει το όρισμά της σε αριθμό κινητής υποδιαστολής. Χρησιμοποιείται εδώ για να εξασφαλίσει ότι το αποτέλεσμα της toFixed(2) (η οποία επιστρέφει μια συμβολοσειρά για τον στρογγυλοποιημένο αριθμό) μετατρέπεται ξανά σε αριθμό κινητής υποδιαστολής.

Ακολουθούνται τα παραπάνω βήματα για να μορφοποιηθούν και να εξαχθούν όλες οι μετρήσεις του παραπάνω πίνακα.

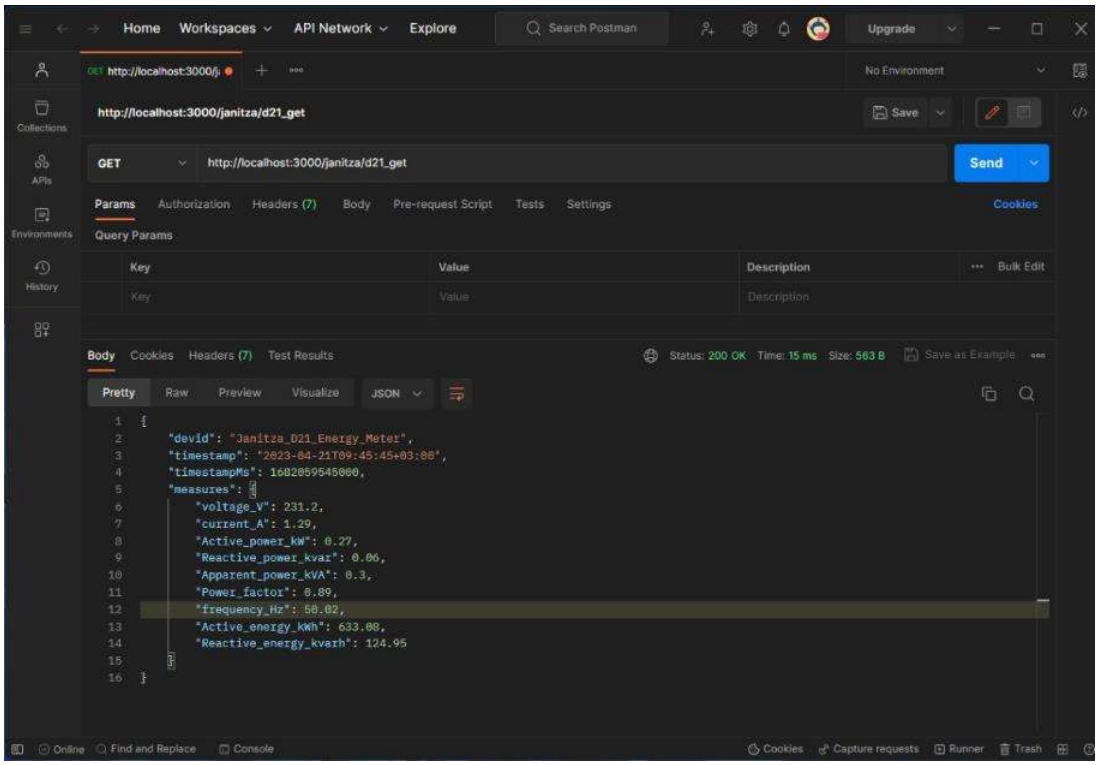
Τελευταίο βήμα αποτελεί η δημιουργία ενός αντικειμένου με όνομα data, το οποίο θα είναι σε μορφή JSON και θα περιέχει όλες τις παραπάνω μετρήσεις μαζί με το αναγνωριστικό της συσκευής και την χρονική ένδειξη (timestamp) για τη λήψη των συγκεκριμένων μετρήσεων. Συνεπώς η συνάρτηση επιστρέφει το αντικείμενο data ως αποτέλεσμα της κλήσης της συνάρτησης GetDataSmartMeterHome(). Εάν προκύψουν σφάλματα κατά τη διάρκεια της διαδικασίας, έχει προβλεφθεί η χρήση του winstonLogger για την καταγραφή των σφαλμάτων.

```
async function GetDataSmartMeterHome(client, reg_start, reg_finish){
    try {
        const now = mom_tz.utc().tz("Europe/Athens").format();
        const timestampMs = Date.parse(now);
        let measure = await client.readHoldingRegisters(reg_start, reg_finish);
        let voltage_V =parseFloat(measure.buffer.readFloatBE(0).toFixed(2));
        let current_A = parseFloat(measure.buffer.readFloatBE(4).toFixed(2));
        let Active_power_kW = parseFloat(measure.buffer.readFloatBE(8).toFixed(2));
        let Reactive_power_kvar = parseFloat(measure.buffer.readFloatBE(12).toFixed(2));
        let Apparent_power_kVA = parseFloat(measure.buffer.readFloatBE(16).toFixed(2));
        let Power_factor = parseFloat(measure.buffer.readFloatBE(20).toFixed(2));
        let frequency_Hz = parseFloat(measure.buffer.readFloatBE(24).toFixed(2));
        let Active_energy_kWh = parseFloat(measure.buffer.readFloatBE(28).toFixed(2));
        let Reactive_energy_kvarh = parseFloat(measure.buffer.readFloatBE(36).toFixed(2));

        const data = {
            devid: "Janitza_D21_Energy_Meter",
            timestamp: now,
            timestampMs: timestampMs,
            measures:  {
                "voltage_V": voltage_V,
                "current_A": current_A,
                "Active_power_kW": Active_power_kW,
                "Reactive_power_kvar": Reactive_power_kvar,
                "Apparent_power_kVA": Apparent_power_kVA,
                "Power_factor": Power_factor,
                "frequency_Hz": frequency_Hz,
                "Active_energy_kWh": Active_energy_kWh,
                "Reactive_energy_kvarh": Reactive_energy_kvarh
            }
        }

        return data;

    } catch (e) {
        winstonLogger.error(e);
        return [];
    }
}
```

Εικόνα 53 Η συνάρτηση GetDataSmartMeterHome() του Janitza D21

4.2.5. Επαλήθευση λειτουργίας μετρητή ενέργειας 'Janitza D21' μέσω Postman

Για να διεξαγάγουμε ελέγχους και να βεβαιωθούμε ότι ο κώδικάς μας λειτουργεί σωστά, θα χρησιμοποιήσουμε το Postman [42], το οποίο αποτελεί ένα δημοφιλές εργαλείο που χρησιμοποιεί πληθώρα προγραμματιστών για τη δοκιμή, την ανάπτυξη και την αντιμετώπιση προβλημάτων (debugging) σε εφαρμογές API (Application Programming Interface). Συνεπώς, για να χρησιμοποιήσουμε το Postman στη δική μας εφαρμογή, θα πρέπει να καλέσουμε τη μέθοδο ‘d21_get’ από το route ‘janitza’ στη διεύθυνση ‘http://localhost:3000/janitza/d21_get’. Το αποτέλεσμα της κλήσης απεικονίζεται στην παρακάτω εικόνα:



Εικόνα 54 Επαλήθευση λειτουργίας μετρητή ενέργειας Janitza D21 μέσω Postman

Η έξοδος της παραπάνω κλήσης της μεθόδου ‘d21_get’ απεικονίζεται στο παρακάτω JSON αρχείο.


```
{
   "devid":"Janitza_D21_Energy_Meter",
   "timestamp":"2023-04-21T09:45:45+03:00",
   "timestampMs":1682059545000,
   "measures":{
      "voltage_V":231.2,
      "current_A":1.29,
      "Active_power_kW":0.27,
      "Reactive_power_kvar":0.06,
      "Apparent_power_kVA":0.3,
      "Power_factor":0.89,
      "frequency_Hz":50.02,
      "Active_energy_kWh":633.08,
      "Reactive_energy_kvarh":124.95
   }
}
```


Εικόνα 55 Έξοδος JSON της μεθόδου ‘d21_get’ του Janitza D21

Στο παραπάνω JSON αρχείο απεικονίζονται τα εξής:

- "devid": "Janitza_D21_Energy_Meter" - Αντιπροσωπεύει το αναγνωριστικό της συσκευής, που προσδιορίζει τον συγκεκριμένο μετρητή ενέργειας ως "Janitza_D21_Energy_Meter".

- "timestamp": "2023-04-21T09:45:45+03:00" - Αυτή η τιμή αναπαριστά τη χρονική στιγμή κατά την οποία ελήφθησαν τα δεδομένα των μετρήσεων. Η μορφή της ώρας ακολουθεί το πρότυπο ISO 8601, με μια μετατόπιση ώρας +03:00 ή +02:00, αναλόγως αν βρισκόμαστε στη θερινή ή χειμερινή ώρα που αντιστοιχεί στη ζώνη ώρας στην οποία ανήκει η Ελλάδα.

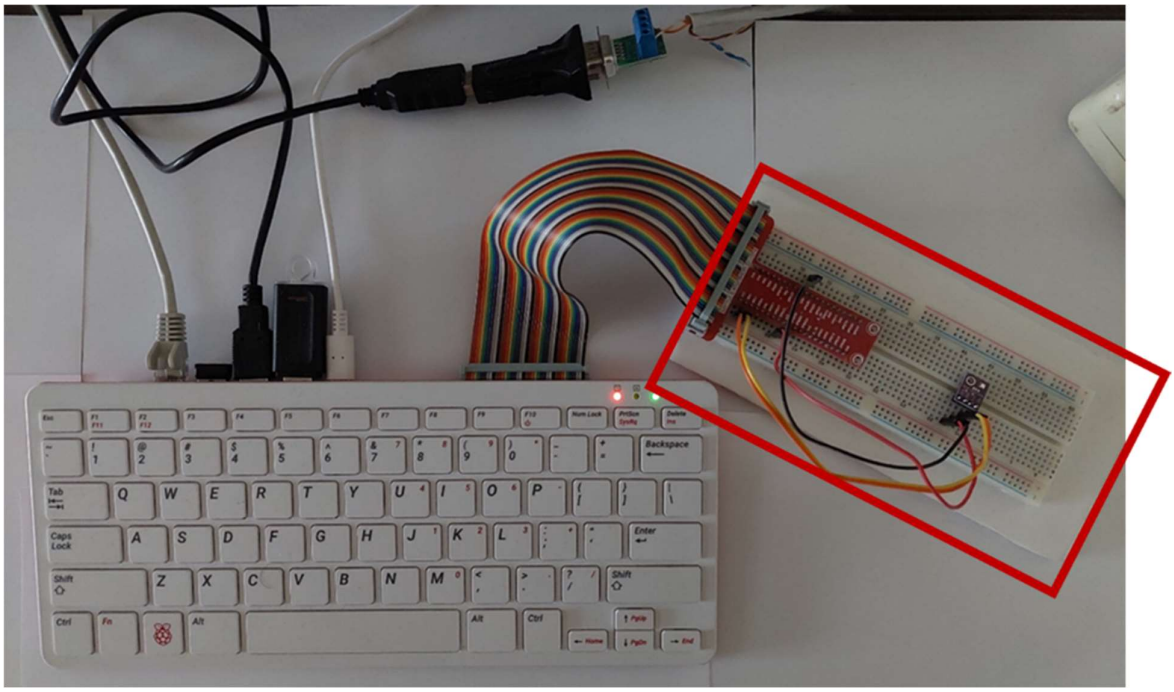
- "timestampMs": 1682059545000 - Πρόκειται για τη χρονική σφραγίδα (timestamp) σε χιλιοστά του δευτερολέπτου από την αρχή Unix time. Ως χρόνος Unix ορίζεται επί του παρόντος, ο αριθμός των δευτερολέπτων που έχουν παρέλθει από τις 00:00:00 UTC της Πέμπτης 1 Ιανουαρίου 1970. Η συγκεκριμένη αναπαράσταση του χρόνου είναι ιδιαίτερα χρήσιμη στα υπολογιστικά συστήματα, καθώς επιτρέπει την εύκολη σύγκριση και τον υπολογισμό των χρονικών διαφορών.

- "measures" - Πρόκειται για ένα αντικείμενο που περιέχει διάφορες μετρήσεις που σχετίζονται με την ένδειξη του μετρητή ενέργειας ‘Janitza D21’:
    - "voltage_V": 231.2 - Η τάση σε volt (V).
    - "current_A": 1.29 - Το ρεύμα σε ampere (A).
    - "Active_power_kW": 0.27 - Η ενεργός ισχύς σε kW.
    - "Reactive_power_kvar": 0.06 - Η άεργος ισχύς σε kvar.
    - "Apparent_power_kVA": 0.3 - Η φαινόμενη ισχύς σε kVA.
    - "Power_factor": 0.89 - Ο συντελεστής ισχύος, ο οποίος είναι ένας καθαρός αριθμός, που αντιπροσωπεύει το λόγο της ενεργού ισχύος προς τη φαινόμενη ισχύ.
    - "frequency_Hz": 50.02 - Η συχνότητα σε Hz.
    - "Active_energy_kWh": 633.08 - Η ενεργός ενέργεια σε κιλοβατώρες σε (kWh).
    - "Reactive_energy_kvarh": 124.95 - Η άεργος ενέργεια σε kvarh.

4.3. Σύνδεση & Ανάλυση του μετρητή περιβάλλοντος “Bosch BME280 sensor”

4.3.1. Σύνδεση του “ Bosch BME280 sensor” στο Raspberry PI

Για να συνδέσουμε τον αισθητήρα BME280 της Bosch στο Raspberry Pi 400, θα πρέπει να δημιουργήσουμε τις απαραίτητες συνδέσεις μεταξύ του αισθητήρα και των ακροδεκτών GPIO (General Purpose Input/Output) του Raspberry Pi χρησιμοποιώντας ένα Raspberry Pi 40 Pin GPIO Ribon Cable Extension Board, μαζί με το GPIO Cobbler Extension Board για τη σύνδεση του Raspberry PI 400 με το Breadboard. Ο αισθητήρας επικοινωνεί με το Raspberry Pi χρησιμοποιώντας το πρωτόκολλο I2C (Inter-Integrated Circuit), όπως φαίνεται στην παρακάτω εικόνα.

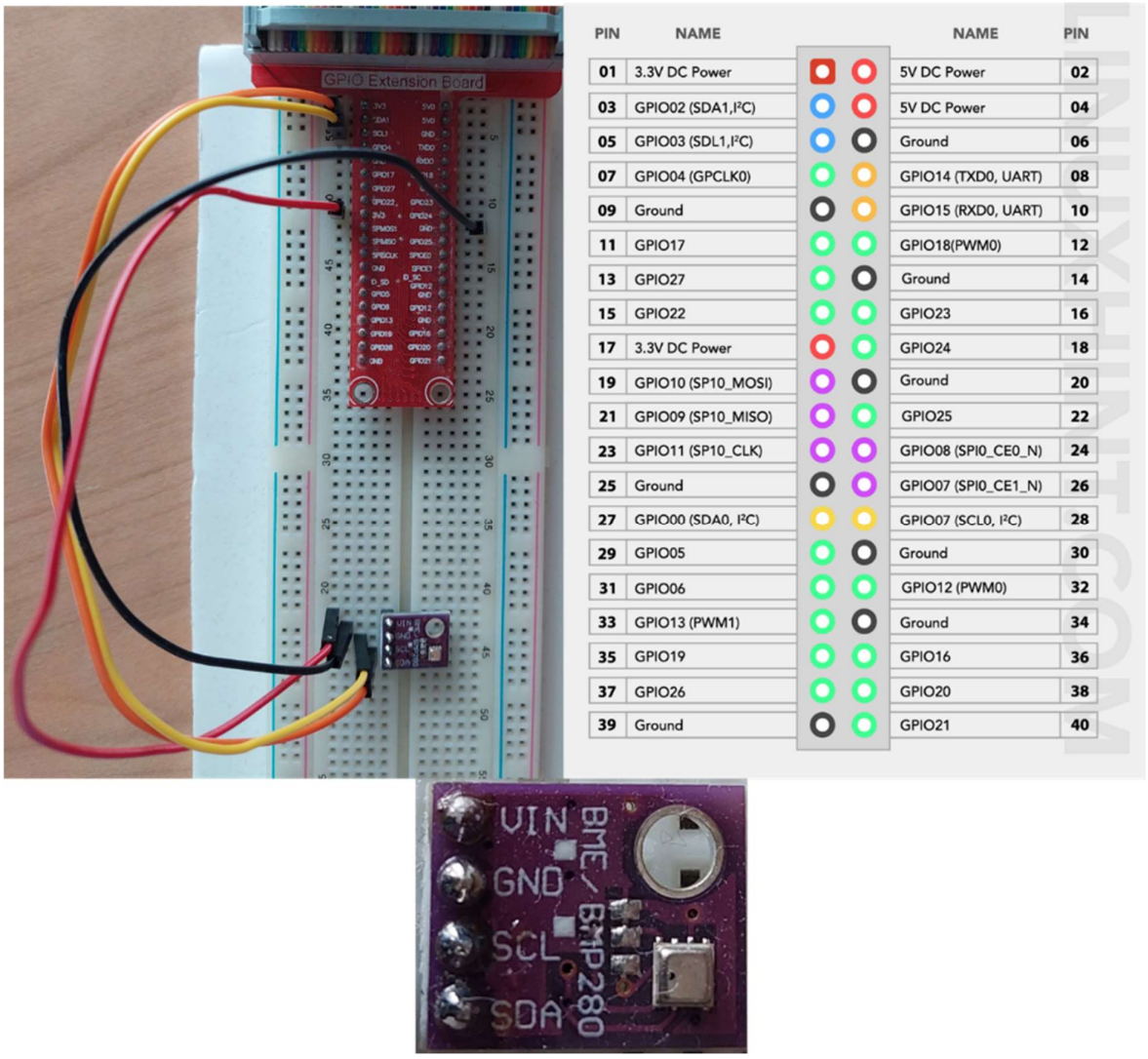


Εικόνα 56 Σύνδεση αισθητήρα BME280 στο Raspberry PI με breadboard

Ο αισθητήρας BME280 αποτελείται από τέσσερις ακροδέκτες: VIN, GND, SDA και SCL.

Ακολουθεί παρακάτω η ανάλυση του κάθε ακροδέκτη:

- VIN (Power Suppl): Αυτός ο ακροδέκτης χρησιμοποιείται για την παροχή ρεύματος στον αισθητήρα BME280. Η τάση τροφοδοσίας που απαιτείται από τον αισθητήρα είναι 3,3V.
- GND (Ground): Ο ακροδέκτης GND συνδέεται με τη γείωση.
- SDA (Serial Data Line): Η SDA είναι μία από τις δύο γραμμές επικοινωνίας στο πρωτόκολλο I2C. Χρησιμοποιείται για την αμφίδρομη σειριακή μεταφορά δεδομένων μεταξύ του αισθητήρα BME280 και του Raspberry Pi 400.
- SCL (Serial Clock Line): Η SCL είναι η δεύτερη γραμμή επικοινωνίας στο πρωτόκολλο I2C. Είναι υπεύθυνη για το συγχρονισμό της μεταφοράς δεδομένων μεταξύ του αισθητήρα BME280 και του Raspberry Pi 400. Ο ακροδέκτης SCL παράγει παλμούς ρολογιού έτσι ώστε να ελέγχει το χρονισμό της μετάδοσης δεδομένων.

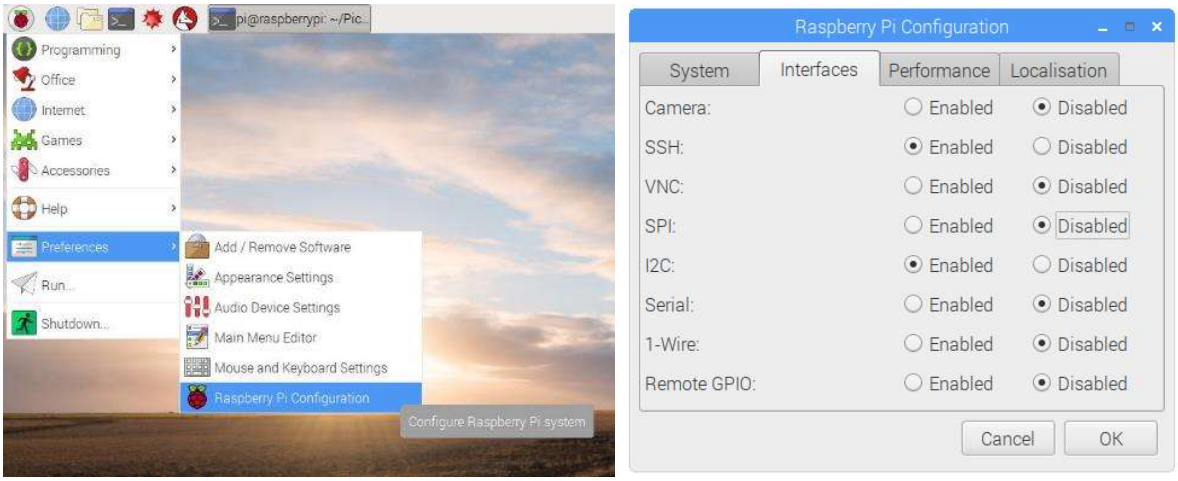


Εικόνα 57 Συνδεσμολογία αισθητήρα BME280 στο Raspberry PI

Αρχικά, θα πρέπει να βεβαιωθούμε ότι το Raspberry Pi 400 είναι απενεργοποιημένο και στη συνέχεια, να συνδέσουμε τους ακροδέκτες του αισθητήρα με το Breadboard, έτσι ώστε να αποφύγουμε τυχόν βραχυκύκλωμα. Έπειτα, θα συνδέσουμε τον ακροδέκτη VCC (τροφοδοσία ρεύματος) του αισθητήρα BME280 με τον ακροδέκτη τροφοδοσίας 3,3V (pin 17 - DC Power) του Raspberry Pi 400 και τον ακροδέκτη GND (γείωση) του αισθητήρα με τον ακροδέκτη 20 (Ground), όπως απεικονίζεται στο παραπάνω σχήμα. Απομένουν οι ακροδέκτες SDA (Serial Data Line) και SCL (Serial Clock Line), οι οποίοι θα συνδεθούν στα pins 03 (GPIO02) και 05 (GPIO03) αντίστοιχα.

4.3.2. Επικοινωνία μετρητή περιβάλλοντος μέσω I2C και Raspberry Pi 4

Μόλις δημιουργηθούν οι συνδέσεις, θα πρέπει να ενεργοποιήσουμε στο Raspberry Pi 400, την επικοινωνία I2C, η οποία βρίσκεται Menu > Preferences > Raspberry Pi Configuration. θα πρέπει να επιλέξετε την καρτέλα "Interfaces" και να ορίσετε το I2C σε "Enabled" όπως απεικονίζεται στις παρακάτω εικόνες.



Εικόνα 58 Ενεργοποίηση I2C στο Raspberry Pi

Τέλος, θα πρέπει να ελέγξουμε ότι είναι εφικτή η σύνδεση I2C μεταξύ του αισθητήρα BME280 και του Raspberry Pi 400, δίνοντας στο τερματικό την παρακάτω εντολή:

```
pi@raspberrypi ~ $ i2cdetect -y 1
```

Εικόνα 59 Έλεγχος σύνδεσης I2C του αισθητήρα BME280 και του Raspberry Pi 400

Η απόκριση της παραπάνω εντολής είναι ότι υπάρχει συνδεδεμένη συσκευή/αισθητήρας με το raspberry Pi και η διεύθυνσή του είναι 0x76 και 118 σε δεκαεξαδικό (hex) και δεκαδικό σύστημα (dec), αντίστοιχα. Τη συγκεκριμένη πληροφορία θα την χρησιμοποιήσουμε παρακάτω, για την υλοποίηση και τη διασύνδεση της βιβλιοθήκης που κατασκευάσαμε, με σκοπό να αντλήσουμε τα ανεπεξέργαστα δεδομένα (raw data) από τον αισθητήρα BME280.

```
pi@raspberrypi ~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

Εικόνα 60 Η διεύθυνσή επικοινωνίας του αισθητήρα με το Raspberry Pi 400

4.3.3. Ανάλυση κώδικα για το μετρητή περιβάλλοντος “Bosch BME280 sensor”

4.3.3.1. Εισαγωγή

Για τη διασύνδεση του μετρητή περιβαλλοντικών παραμέτρων με το Raspberry PI 4 και την εξαγωγή των δεδομένων, δημιουργήθηκε ο παρακάτω κώδικας σε node.js. Ο αναλυτικός κώδικας μπορεί να βρεθεί στο παράρτημα, στο τέλος της διπλωματικής εργασίας. Στο παρόν τμήμα, θα αναλύσουμε τα κυριότερα σημεία του κώδικα που είναι υπεύθυνα για την άντληση των δεδομένων.

4.3.3.2. Αρχικοποίηση RabbitMQ

Στην αρχή του παρακάτω κώδικα, δημιουργήθηκε το αντικείμενο RABBIT_PARAMS που περιέχει την κατάλληλή παραμετροποίηση για τη σύνδεση και την επικοινωνία με το διαμεσολαβητή μηνυμάτων (message broker) RabbitMQ. Το αντικείμενο RABBIT_PARAMS έχει αναλυθεί εκτενέστερα για τον ενεργειακό αναλυτή ‘Janitza D21’. Η τροποποίηση που θα χρειαστούμε σε σχέση με τον ενεργειακό αναλυτή, είναι να δημιουργήσουμε ένα νέο κλειδί δρομολόγησης για να φιλτράρονται τα δεδομένα που θα συλλέγονται από τον αισθητήρα BME280. Το κλειδί δρομολόγησης που χρησιμοποιούμε είναι **Bme280SensorRoutingKey. Η υπόλοιπη λειτουργία του** message broker RabbitMQ θα είναι όπως και στον ενεργειακό αναλυτή.

```
const RABBIT_PARAMS = {
    HOST: config.RABBIT.HOST,
    PORT: config.RABBIT.PORT,
    USER: config.RABBIT.SMART_USER,
    PASSWORD: config.RABBIT.SMART_PASSWORD,
    ExchangeName: 'amq.topic',
    ExchnageType: 'topic',
    Bme280SensorRoutingKey: 'bme280.master',
    IsDurable: true
}
```

Εικόνα 61 Αρχικοποίηση μετρητή περιβάλλοντος BME280 στο RabbitMQ

Ακόμη, κάνοντας χρήση της βιβλιοθήκης 'i2c-bus' έχουμε τη δυνατότητα να αλληλεπιδράσουμε και να αντλήσουμε δεδομένα από τον αισθητήρα BME280 μέσω του πρωτοκόλλου επικοινωνίας I2C (Inter-Integrated Circuit). Για την εγκαθίδρυση σύνδεσης θα πρέπει να ορίσουμε την πόρτα (port) και τη διεύθυνση (address) του διαύλου I2C, τα οποία έχουν οριστεί σε 1 και 0x76, αντίστοιχα.

4.3.3.3. Κλήση μεθόδου ‘bme280_get’ (HTTP GET request)

Χρησιμοποιώντας το framework Express.js και τη μέθοδο router.get() για τον καθορισμό ενός χειριστή διαδρομής για εισερχόμενα αιτήματα HTTP GET. Με άλλα λόγια, χρησιμοποιώντας τη συγκεκριμένη μέθοδο, καλούμε ένα HTTP GET request είτε με τη βοήθεια του Postman είτε ενός browser, με σκοπό τον έλεγχο ή την αποσφαλμάτωση (debugging) του κώδικα. Τη συγκεκριμένη λειτουργεία για τον αισθητήρα BME280 θα τη γνωρίσουμε στην επόμενη ενότητα.

Ακολουθεί ο block κώδικας (Try-catch), όπου εμπεριέχεται η κυρίως λειτουργία της εφαρμογής. Χρησιμοποιήθηκε η βιβλιοθήκη ‘winstonLogger’ η οποία μας βοηθάει στην καταγραφή των logs τόσο κατά τον εντοπισμό σφαλμάτων (erros) winstonLogger.error() όσο και κατά τη διάρκεια της αποσφαλμάτωσης (debug). Στις επόμενες γραμμές, ορίζουμε το όνομα του virtual_host = 'smart_meters' έτσι όπως το έχουμε ορίσει στο διαμεσολαβητή RabbitMQ, ενώ χρησιμοποιούμε τη βιβλιοθήκη ‘moment-timezone’ για να μετατρέψουμε την τρέχουσα ημερομηνία και ώρα (timestamp) στη ζώνη ώρας "Europe/Athens" σε χιλιοστά του δευτερολέπτου. Μετά, ακολουθεί ο έλεγχος για το εάν μπορεί να πραγματοποιηθεί σύνδεση με το διαμεσολαβητή, λαμβάνοντας ως ορίσματα το όνομα του παραπάνω virtual_host και των παραμέτρων RABBIT_PARAMS. Εάν η σύνδεση δεν μπορεί να επιτευχθεί, τότε εμφανίζεται το μήνυμα “ No Rabbit channel available !” και τερματίζεται η εκτέλεση του κώδικα. Μόλις η σύνδεση με το διαμεσολαβητή επιτευχθεί, καλείται η συνάρτηση bme280Init η οποία αρχικοποιεί τον αισθητήρα και επιστρέφει τις απαραίτητες παραμέτρους οι οποίες θα χρησιμοποιηθούν για την κλήση της συνάρτησης bme280Start. Η συγκεκριμένη συνάρτηση θα επιστρέψει τα ανεπεξέργαστα δεδομένα (raw data) του αισθητήρα. Η λειτουργία των συναρτήσεων bme280Init bme280Start θα αναλυθεί παρακάτω.

Τελευταίο βήμα αποτελεί η δημιουργία ενός JSON αντικειμένου, με όνομα json, που περιλαμβάνει τις μετρήσεις του αισθητήρα (θερμοκρασία, υγρασία, πίεση), τον κωδικό της συσκευής και τη χρονική ένδειξη (timestamp) για τη λήψη των συγκεκριμένων μετρήσεων. Αφού τα δεδομένα δημιουργηθούν και προστεθούν στο αντικείμενο json, το μήνυμα στέλνεται στο RabbitMQ μέσω της μεθόδου SendToRabbitmq(), χρησιμοποιώντας τις απαραίτητες παραμέτρους που αναφέραμε προηγουμένως. Αποστέλλουμε τα δεδομένα της μεταβλητής json στον πελάτη – χρήστη, ως απάντηση (response) του HTTP GET request αιτήματος που υπέβαλε προς την εφαρμογή.

Εάν προκύψουν σφάλματα κατά τη διάρκεια της διαδικασίας, έχει προβλεφθεί η χρήση του winstonLogger για την καταγραφή των σφαλμάτων.

```
const port = 1;
const address = 0x76;
const bus = i2c.openSync(port);

router.get('/bme280_get', async function (req, res, next) {
    try {
        winstonLogger.log('debug', req.query);
        const virtual_host = 'smart_meters';
        const channel = await rabbit.GetChannel(virtual_host, RABBIT_PARAMS);

        const now = mom_tz.utc().tz("Europe/Athens").format();
        const timestampMs = Date.parse(now);

        if (!channel) {
          res.send("No Rabbit channel available !");
          return;
      }

      const initParameters = await bme280Init();
      const data = await bme280Start(initParameters);

        const json =  {
              devid: "bme280_sensor",
              timestamp: now,
              timestampMs: timestampMs,
              measures:  {
                  "temperature": data.temperature.toFixed(2),
                  "humidity": data.humidity.toFixed(2),
                  "pressure": data.pressure.toFixed(2)
              }
            };

      rabbit.SendToRabbitmq(channel, RABBIT_PARAMS.ExchangeName, RAB-
BIT_PARAMS.Bme280SensorRoutingKey, json);
      res.send(json);

    } catch (e) {
      winstonLogger.error(e);
      res.send(e);
    }
  });
```

Εικόνα 62 Κλήση της μεθόδου bme280_get του μετρητή περιβάλλοντος BME280

4.3.3.4. Η συνάρτηση bme280Init()

Με τη χρήση της συγκεκριμένης συνάρτησης αρχικοποιούνται οι καταχωρητές για τον αισθητήρα BME280. Οι συγκεκριμένοι καταχωρητές απεικονίζονται στο datasheet [41] του αισθητήρα με την ονομασία calibration data, όνομα καταχωρητή calib00 – calib25, διεύθυνση μνήμης 0x88 – 0x41 και όνομα καταχωρητή calib26 – calib41, διεύθυνση μνήμης 0xE1 – 0xF0.

Οι συγκεκριμένοι καταχωρητές θα χρησιμοποιηθούν για την άντληση των δεδομένων, απεικονίζονται συνοπτικό στον παρακάτω πίνακα.

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
hum_lsb	0xFE	hum_lsb<7:0>								0x00
hum_msb	0xFD	hum_msb<7:0>								0x80
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]			filter[2:0]				spi3w_en[0]	0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]		0x00
status	0xF3					measuring[0]			im_update[0]	0x00
ctrl_hum	0xF2						osrs_h[2:0]			0x00
calib26..calib41	0xE1…0xF0	calibration data								individual
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x60
calib00..calib25	0x88…0xA1	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Chip ID	Reset
Type:	do not change	read only	read / write	read only	read only	read only	write only

Εικόνα 63 Οι καταχωρητές (registers) του μετρητή περιβάλλοντος BME280

Έπειτα, στον πίνακα που ακολουθεί περιγράφονται αναλυτικά ο τύπος δεδομένων του κάθε καταχωρητή καθώς και οι διευθύνσεις μνήμης που θα πρέπει να προσπελαστούν.

Πίνακας 7 Οι καταχωρητές (registers) του BME280 με τις διευθύνσεις μνήμης

Register Address	**Register content**	**Data type**
0x88 / 0x89	dig_T1 [7:0] / [15:8]	unsigned short
0x8A / 0x8B	dig_T2 [7:0] / [15:8]	signed short
0x8C / 0x8D	dig_T3 [7:0] / [15:8]	signed short
0x8E / 0x8F	dig_P1 [7:0] / [15:8]	unsigned short
0x90 / 0x91	dig_P2 [7:0] / [15:8]	signed short
0x92 / 0x93	dig_P3 [7:0] / [15:8]	signed short
0x94 / 0x95	dig_P4 [7:0] / [15:8]	signed short
0x96 / 0x97	dig_P5 [7:0] / [15:8]	signed short
0x98 / 0x99	dig_P6 [7:0] / [15:8]	signed short
0x9A / 0x9B	dig_P7 [7:0] / [15:8]	signed short
0x9C / 0x9D	dig_P8 [7:0] / [15:8]	signed short
0x9E / 0x9F	dig_P9 [7:0] / [15:8]	signed short
0xA1	dig_H1 [7:0]	unsigned char
0xE1 / 0xE2	dig_H2 [7:0] / [15:8]	signed short
0xE3	dig_H3 [7:0]	unsigned char
0xE4 / 0xE5[3:0]	dig_H4 [11:4] / [3:0]	signed short
0xE5[7:4] / 0xE6	dig_H5 [3:0] / [11:4]	signed short

0xE7	dig_H6	signed char

```
async function bme280Init() {
  const calib = {};

  // Temperature trimming params
  calib['dig_T1'] = unsignedShort(0x88);
  calib['dig_T2'] = signedShort(0x8A);
  calib['dig_T3'] = signedShort(0x8C);

  // Pressure trimming params
  calib['dig_P1'] = unsignedShort(0x8E);
  calib['dig_P2'] = signedShort(0x90);
  calib['dig_P3'] = signedShort(0x92);
  calib['dig_P4'] = signedShort(0x94);
  calib['dig_P5'] = signedShort(0x96);
  calib['dig_P6'] = signedShort(0x98);
  calib['dig_P7'] = signedShort(0x9A);
  calib['dig_P8'] = signedShort(0x9C);
  calib['dig_P9'] = signedShort(0x9E);

  // Humidity trimming params
  calib['dig_H1'] = unsignedByte(0xA1);
  calib['dig_H2'] = signedShort(0xE1);
  calib['dig_H3'] = unsignedByte(0xE3);
  calib['dig_H4'] = (signedByte(0xE4) << 4) | (signedByte(0xE5) & 0x0F);
  calib['dig_H5'] = ((signedByte(0xE5) >> 4) & 0x0F) | (signedByte(0xE6) << 4);
  calib['dig_H6'] = signedByte(0xE7);

  return calib;
}
```

Εικόνα 64 Αρχικοποίηση καταχωρητών για το μετρητή περιβάλλοντος BME280

4.3.3.5. Η συνάρτηση bme280Start()

Η συνάρτηση bme280Start() χρησιμοποιείται για την μέτρηση και καταγραφή των μετρήσεων χρησιμοποιώντας τον αισθητήρα BME280, ο οποίος μπορεί να συλλέξει θερμοκρασία, υγρασία και ατμοσφαιρική πίεση. Θα μπορούσαμε να χαρακτηρίσουμε τη συγκεκριμένη συνάρτηση, τη main συνάρτηση του αισθητήρα BME280.

Ο αισθητήρας BME280 προσφέρει τρεις τρόπους λειτουργίας σύμφωνα με το datasheet [41]. Η λειτουργία που θα επιλεγεί εξαρτάται από την εφαρμογή και το ρυθμό που θέλουμε να αντλούμε μετρήσεις από τη συσκευή. Οι τρεις λειτουργίες που προσφέρει ο αισθητήρας είναι:

- **Sleep mode**: Σε αυτή τη λειτουργία η συσκευή, βρίσκεται σε αδράνεια και ουσιαστικά απενεργοποιείται. Δεν πραγματοποιούνται μετρήσεις, γεγονός που ελαχιστοποιεί την κατανάλωση ενέργειας.
- **Forced mode**: Στην συγκεκριμένη λειτουργία η συσκευή, πραγματοποιεί μία μόνο μέτρηση της θερμοκρασίας, της υγρασίας και της ατμοσφαιρικής πίεσης και στη συνέχεια επιστρέφει στην κατάσταση αναστολής λειτουργίας (sleep mode).
- **Normal mode**: Στην Κανονική λειτουργία, η συσκευή εναλλάσσεται συνεχώς μεταξύ μιας ενεργής περιόδου μέτρησης και μιας περιόδου αναμονής. Αυτή η λειτουργία είναι κατάλληλη αν χρειάζεται να λαμβάνεται μετρήσεις συνεχώς από τη συσκευή.

Η λειτουργία που έχουμε επιλέξει για την άντληση των δεδομένων της εργασίας είναι forced mode.

Επιπλέον, θα πρέπει να ορίσουμε σύμφωνα με το datasheet [41] εκτός του τρόπου λειτουργίας του αισθητήρα και το ρυθμό δειγματοληψίας, έτσι ώστε να αποφύγουμε θόρυβο κατά τη λήψη της μέτρησης που θα έχει ως αποτέλεσμα την αλλοίωση της. Ο ρυθμός δειγματοληψίας ορίζεται από την OVERSAMPLING_X1 για κάθε τύπο μέτρησης (θερμοκρασία, υγρασία, ατμοσφαιρική πίεση) και στη συνέχεια πραγματοποιείται εγγραφή στους κατάλληλους καταχωρητές για να ενημερώσουν τον αισθητήρα τόσο για τη λειτουργία που θα πρέπει να επιλέξει, όσο και για το ρυθμό δειγματοληψίας. Οι συγκεκριμένοι καταχωρητές που θα πρέπει να ενημερωθούν είναι οι ctrl_meas (0xF4) που καθορίζουν τον τρόπου που θα γίνει η συλλογή των μετρήσεων θερμοκρασίας, υγρασίας και ctrl_hum (0xF2) που καθορίζει τον τρόπου που θα γίνει η συλλογή της υγρασίας.

```
  async function bme280Start(params) {
    const mode = 1;  // forced
    const tOversampling = OVERSAMPLING_X1;
    const hOversampling = OVERSAMPLING_X1;
    const pOversampling = OVERSAMPLING_X1;

    // ctrl_hum
    bus.writeByteSync(address, 0xF2, hOversampling);
    // ctrl_meas
    bus.writeByteSync(address, 0xF4, tOversampling << 5 | pOversampling << 2 |
mode);
    await sleep(10);

    const buffer = Buffer.alloc(8);
    bus.readI2cBlockSync(address, 0xF7, 8, buffer);
    const raw_data = rawMeasurements(buffer);

    const data_readings = calcMeasurements(raw_data, params);
    return data_readings;
  }
```

Εικόνα 65 Μέτρηση και καταγραφή των μετρήσεων με τη συνάρτηση bme280Start()

4.3.3.6. Η συνάρτηση rawMeasurements()

Η συνάρτηση rawMeasurements() λαμβάνει ένα block δυαδικών δεδομένων (block of binary data), από τον αισθητήρα BME280 και με κατάλληλους υπολογισμούς διαχωρίζει τα ανεπεξέργαστα δεδομένα (raw data). Για να πραγματοποιηθεί ο συγκεκριμένος διαχωρισμός θα πρέπει να εκτελεστούν ξεχωριστοί υπολογισμοί για θερμοκρασία, υγρασία και ατμοσφαιρική πίεση. Σύμφωνα με το datasheet [41] του αισθητήρα, οι καταχωρητές ατμοσφαιρικής πίεσης και υγρασίας αποτελούνται από 24 bits (bit0-bit3 είναι 0), ενώ οι καταχωρητές της υγρασίας από 16 bits. Τα bits από bit0 έως bit3 είναι 0, για τους καταχωρητές ατμοσφαιρικής πίεσης και υγρασίας, επομένως χρησιμοποιούνται μόνο τα 20 bits στους συγκεκριμένους. Παρακάτω απεικονίζονται αναλυτικά οι συγκεκριμένοι καταχωρητές.

Πίνακας 9 Καταχωρητές ατμοσφαιρικής πίεσης BME280

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80

Register 0xF7...0xF9 *"press"*	Name	Description
0xF7	press_msb[7:0]	Contains the MSB part up[19:12] of the raw pressure measurement output data.
0xF8	press_lsb[7:0]	Contains the LSB part up[11:4] of the raw pressure measurement output data.
0xF9 (bit 7, 6, 5, 4)	press_xlsb[3:0]	Contains the XLSB part up[3:0] of the raw pressure measurement output data. Contents depend on temperature resolution.

Πίνακας 8 Καταχωρητές θερμοκρασίας BME280

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80

Register 0xFA...0xFC *"temp"*	Name	Description
0xFA	temp_msb[7:0]	Contains the MSB part ut[19:12] of the raw temperature measurement output data.
0xFB	temp_lsb[7:0]	Contains the LSB part ut[11:4] of the raw temperature measurement output data.
0xFC (bit 7, 6, 5, 4)	temp_xlsb[3:0]	Contains the XLSB part ut[3:0] of the raw temperature measurement output data. Contents depend on pressure resolution.

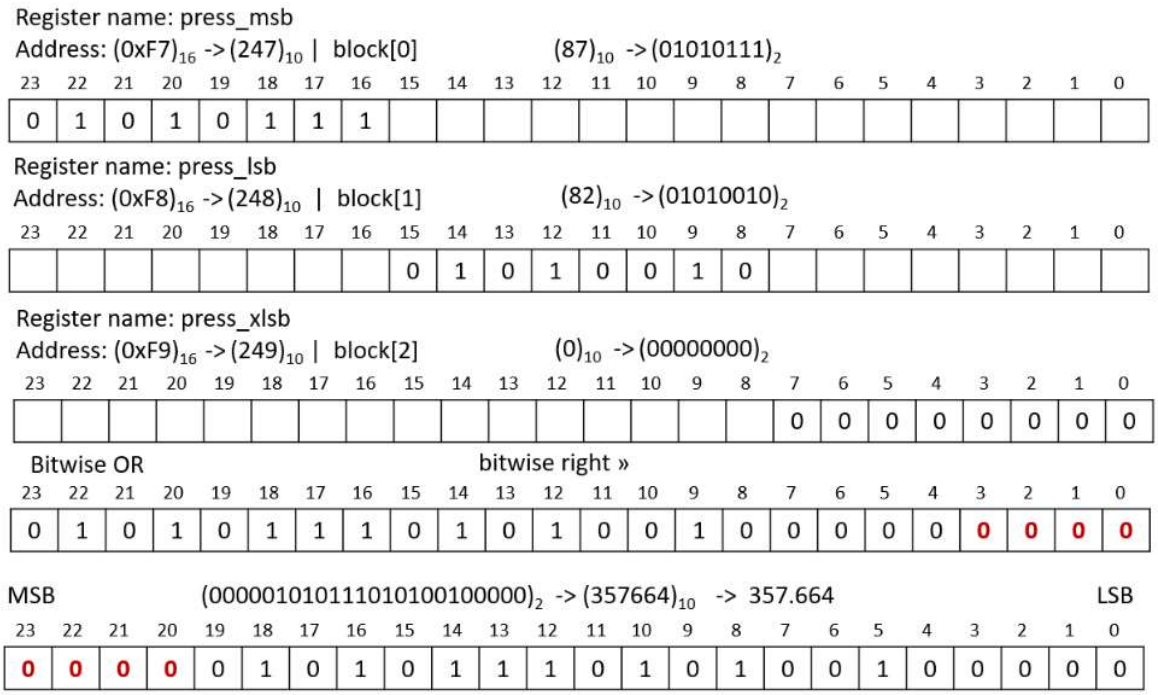
Πίνακας 10 Καταχωρητές υγρασίας BME280

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
hum_lsb	0xFE	hum_lsb<7:0>								0x00
hum_msb	0xFD	hum_msb<7:0>								0x80

Register 0xFD...0xFE *"hum"*	Name	Description
0xFD	hum_msb[7:0]	Contains the MSB part uh[15:8] of the raw humidity measurement output data.
0xFE	temp_lsb[7:0]	Contains the LSB part uh[7:0] of the raw humidity measurement output data.

Για να γίνει ευκολότερα κατανοητή η λειτουργεία της συγκεκριμένης συνάρτησης, στους παρακάτω πίνακες, απεικονίζονται τα δεδομένα με τις πράξεις που εκτελέστηκαν κατά τη διάρκεια δοκιμών της συνάρτησης (debugging). Στο συγκεκριμένο παράδειγμα, απεικονίζεται ο υπολογισμός της ατμοσφαιρικής πίεσης. Με τον ίδιο τρόπο υπολογίζεται η θερμοκρασία και με ελάχιστες αλλαγές η υγρασία.

Αρχικά, σύμφωνα με τον κώδικα, μετατοπίζονται αριστερά (bitwise left «) τα bits από το block[0] κατά 16 θέσεις, τα bits από το block[1] κατά 8 θέσεις, ενώ τα bits από το block[2] δεν μετατοπίζονται. Για να προκύψει ο τελικός πίνακας, θα πρέπει να εκτελεστεί ο τελεστής | (bitwise OR operation) και στο αποτέλεσμα που θα προκύψει, να γίνει τελική μετατόπιση δεξιά (bitwise right ») κατά 4 θέσεις. Η παραπάνω λειτουργία, απεικονίζεται παρακάτω:



Εικόνα 66 Παράδειγμα υπολογισμού ατμοσφαιρικής πίεσης στους καταχωρητές

```
function rawMeasurements(block) {
  const pressure = (block[0] << 16 | block[1] << 8 | block[2]) >> 4;
  const temperature = (block[3] << 16 | block[4] << 8 | block[5]) >> 4;
  const humidity = block[6] << 8 | block[7];

  return {
    temperature: temperature,
    pressure: pressure,
    humidity: humidity
  };
}
```

Εικόνα 67 Η συνάρτηση rawMeasurements()

4.3.3.7. Η συνάρτηση calcMeasurements()

Η συνάρτηση calcMeasurements() εκτελεί υπολογισμούς στα ακατέργαστα δεδομένα (raw data), που έχουν αντληθεί από τον αισθητήρα BME280, έτσι ώστε να σταλούν σε κατάλληλη μορφή στον χρήστη. Στη συγκεκριμένη συνάρτηση υπολογίζονται οι τιμές θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης. Ο συγκεκριμένος κώδικας έχει αντληθεί από το παράρτημα Α, σελίδα 49 του datasheet BME280 της Bosch [41].

```
function calcMeasurements(rawReadings, compensationParams) {

    let temperature = 0.0, humidity = 0.0, pressure = 0.0;

    // Calculate temperature
    const v1 = (rawReadings.temperature / 16384.0 - compensationParams.dig_T1 /
1024.0) * compensationParams.dig_T2;
    const v2 = (Math.pow(rawReadings.temperature / 131072.0 - compensationPar-
ams.dig_T1 / 8192.0, 2)) * compensationParams.dig_T3;
    temperature = (v1 + v2) / 5120.0;

    // Calculate humidity
    let res = temperature - 76800.0;
    res = (rawReadings.humidity - (compensationParams.dig_H4 * 64.0 + compensa-
tionParams.dig_H5 / 16384.0 * res)) * (compensationParams.dig_H2 / 65536.0 * (1.0
+ compensationParams.dig_H6 / 67108864.0 * res * (1.0 + compensationParams.dig_H3
/ 67108864.0 * res)));
    res = res * (1.0 - compensationParams.dig_H1 * res / 524288.0);
    humidity = Math.max(0.0, Math.min(res, 100.0));
```

Εικόνα 68 Η συνάρτηση calcMeasurements(), μέρος πρώτο

```
    // Calculate pressure
    let v1p = temperature / 2.0 - 64000.0;
    let v2p = v1p * v1p * compensationParams.dig_P6 / 32768.0;
    v2p = v2p + v1p * compensationParams.dig_P5 * 2.0;
    v2p = v2p / 4.0 + compensationParams.dig_P4 * 65536.0;
    v1p = (compensationParams.dig_P3 * v1p * v1p / 524288.0 + compensationPar-
ams.dig_P2 * v1p) / 524288.0;
    v1p = (1.0 + v1p / 32768.0) * compensationParams.dig_P1;

    // Prevent divide by zero
    if (v1p === 0) {
      res = 0;
    } else {
      res = 1048576.0 - rawReadings.pressure;
      res = ((res - v2p / 4096.0) * 6250.0) / v1p;
      v1p = compensationParams.dig_P9 * res * res / 2147483648.0;
      v2p = res * compensationParams.dig_P8 / 32768.0;
      res = res + (v1p + v2p + compensationParams.dig_P7) / 16.0;
      pressure = res / 100.0;
    }

    return {
      temperature: temperature,
      humidity: humidity,
      pressure: pressure,
    };
  }
```

Εικόνα 69 Η συνάρτηση calcMeasurements(), μέρος δεύτερο

4.3.3.8. Βοηθητικές συναρτήσεις

Ακολουθούν οι βοηθητικές συναρτήσεις για την ανάγνωση των δεδομένων του αισθητήρα χρησιμοποιώντας το πρωτόκολλο I2C:

- **unsignedShort(register**): Η συνάρτηση unsignedShort διαβάζει μια τιμή 16 bit (ή 2 bytes) από έναν συγκεκριμένο καταχωρητή του αισθητήρα BME280 μέσω του πρωτόκολλο I2C και την επιστρέφει ως ακέραιο αριθμό χωρίς πρόσημο (unsigned integer) στο εύρος τιμών από 0 έως 65536 (2^16). Αυτό προκύπτει επειδή καλείται η bus.readWordSync(address, register) με address = 0x76 και register τον καταχωρητή

που μεταβιβάζεται ως παράμετρος από την κλήση της συνάρτησης unsignedShort. Επιπλέον, για να διασφαλίσουμε ότι τα δεδομένα θα είναι σωστά μορφοποιημένα χρησιμοποιούμε τον τελεστή bitwise AND με τον δεκαεξαδικό αριθμό 0xffff (**& 0xffff),** ο οποίος είναι ισοδύναμος με τον δυαδικό αριθμό 1111111111111111 (16 bits), έτσι ώστε να είμαστε σίγουροι ότι τα όλα τα bit πέραν των χαμηλότερων 16 θα απορρίπτονται.

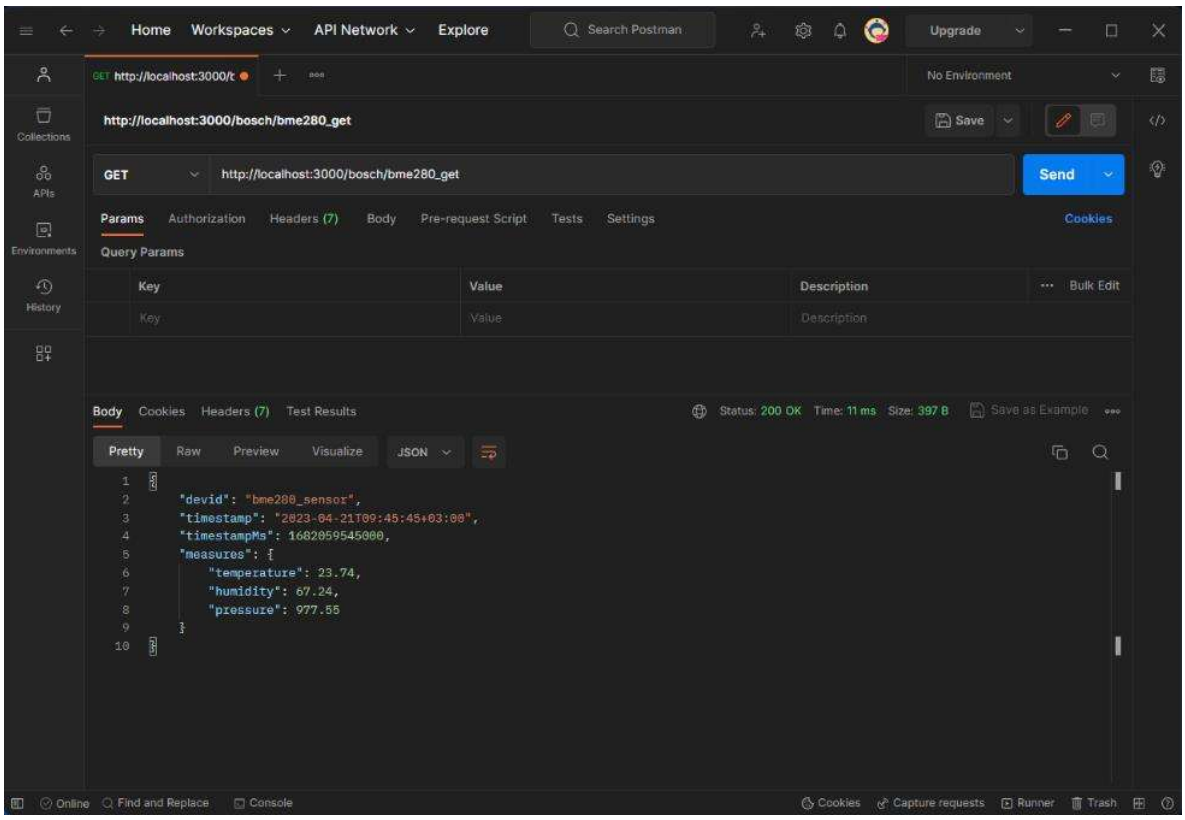
- **unsignedByte(register**): Η συνάρτηση unsignedByte διαβάζει μια τιμή 8 bit (ή 1 byte) από έναν συγκεκριμένο καταχωρητή του αισθητήρα BME280 μέσω του πρωτόκολλο I2C και την επιστρέφει ως ακέραιο αριθμό χωρίς πρόσημο (unsigned integer) στο εύρος τιμών από 0 έως 255 (2^8). Αυτό προκύπτει επειδή καλείται η bus.readByteSync(address, register) με address = 0x76 και register τον καταχωρητή που μεταβιβάζεται ως παράμετρος από την κλήση της συνάρτησης unsignedByte. Επιπλέον, για να διασφαλίσουμε ότι τα δεδομένα θα είναι σωστά μορφοποιημένα χρησιμοποιούμε τον τελεστή bitwise AND με τον δεκαεξαδικό αριθμό 0xff (**& 0xff),** ο οποίος είναι ισοδύναμος με τον δυαδικό αριθμό 11111111 (8 bits), έτσι ώστε να είμαστε σίγουροι ότι τα όλα τα bit πέραν των χαμηλότερων 8 θα απορρίπτονται.
- **signedShort(register):** Η συνάρτηση signedShort λειτουργεί όπως η unsignedShort που αναλύσαμε παραπάνω, με τη διαφορά ότι εκτελεί έναν επιπλέον υπολογισμό πριν την επιστροφή τιμής. Εάν η τιμή της word είναι μικρότερη από 0x8000 (32.768 σε δεκαδικό σύστημα, 2^15), η συνάρτηση επιστρέφει τη word ως έχει, επειδή βρίσκεται εντός του εύρους ενός προσημασμένου ακέραιου αριθμού (signed integer) 16 bit. Ωστόσο, εάν η τιμή της word είναι 0x8000 ή μεγαλύτερη, αυτό σημαίνει ότι η word αντιπροσωπεύει έναν αρνητικό αριθμό, οπότε η συνάρτηση αφαιρεί 0x10000 (65.536 σε δεκαδικό, 2^16) από τη word για να τη μετατρέψει σε σωστή αρνητική τιμή.
- **signedByte(register):** Η συνάρτηση **signedByte** λειτουργεί όπως η **unsignedByte** που αναλύσαμε παραπάνω, με τη διαφορά ότι εκτελεί έναν επιπλέον υπολογισμό πριν την επιστροφή τιμής. Εάν η τιμή της byte είναι μικρότερη από 0x80 (128 σε δεκαδικό σύστημα, 2^7), η συνάρτηση επιστρέφει τη byte ως έχει, επειδή το byte αντιπροσωπεύει θετικό αριθμό (ή μηδέν) και επιστρέφεται ως έχει. Ωστόσο, εάν η τιμή της byte είναι 0x80 ή μεγαλύτερη, αυτό σημαίνει ότι η byte αντιπροσωπεύει έναν αρνητικό αριθμό, οπότε η συνάρτηση αφαιρεί 0x100 (256 σε δεκαδικό, 2^8) από τη byte για να τη μετατρέψει σε σωστή αρνητική τιμή.

```
function unsignedShort(register) {
  return bus.readWordSync(address, register) & 0xffff;
}

function unsignedByte(register) {
  return bus.readByteSync(address, register) & 0xff;
}

function signedShort(register) {
  const word = unsignedShort(register);
  if (word < 0x8000) {
    return word;
  } else {
      return word - 0x10000;
  }
}

function signedByte(register) {
  const byte = unsignedByte(register);
  if (byte < 0x80) {
    return byte;
  } else {
      return byte - 0x100;
  }
}
```

Εικόνα 70 Βοηθητικές συναρτήσεις για ανάγνωση δεδομένων

4.4. Επαλήθευση λειτουργίας μετρητή περιβάλλοντος 'Bosch BME280' μέσω Postman

Για να διεξαγάγουμε ελέγχους και να βεβαιωθούμε ότι ο κώδικάς που δημιουργήσαμε λειτουργεί αποτελεσματικά και χωρίς σφάλματα, θα χρησιμοποιήσουμε το λογισμικό Postman [42], όπως και με το μετρητή ενέργειας "Janitza D21" σε προηγούμενη παράγραφο. Συνεπώς, για να χρησιμοποιήσουμε το Postman, θα πρέπει να καλέσουμε τη μέθοδο '_get' από το route 'bosch' στη διεύθυνση 'http://localhost:3000/bosch/bme280_get'. Το αποτέλεσμα της κλήσης απεικονίζεται στην παρακάτω εικόνα:



Εικόνα 71 Επαλήθευση λειτουργίας μετρητή περιβάλλοντος μέσω Postman

Η έξοδος της παραπάνω κλήσης της μεθόδου 'bme280_get' απεικονίζεται στο παρακάτω JSON αρχείο.

```
{
   "devid":"bme280_sensor",
   "timestamp":"2023-04-21T09:45:45+03:00",
   "timestampMs":1682059545000,
   "measures":{
      "temperature":23.74,
      "humidity":67.24,
      "pressure":977.55
   }
}
```


Εικόνα 72 Έξοδος JSON της μεθόδου ‘bme280_get’ του Bosch BME280

Στο παραπάνω JSON αρχείο απεικονίζονται τα εξής:

- "devid": "bme280_sensor”- Αντιπροσωπεύει το αναγνωριστικό της συσκευής, που προσδιορίζει τον συγκεκριμένο μετρητή περιβαλλοντικών παραμέτρων ως "bme280_sensor ".

- "timestamp": "2023-04-21T09:45:45+03:00" - Αυτή η τιμή αναπαριστά τη χρονική στιγμή κατά την οποία ελήφθησαν τα δεδομένα των μετρήσεων. Η μορφή της ώρας ακολουθεί το πρότυπο ISO 8601, με μια μετατόπιση ώρας +03:00 ή +02:00, αναλόγως αν βρισκόμαστε στη θερινή ή χειμερινή ώρα που αντιστοιχεί στη ζώνη ώρας στην οποία ανήκει η Ελλάδα.

- "timestampMs": 1682059545000 - Πρόκειται για τη χρονική σφραγίδα (timestamp) σε χιλιοστά του δευτερολέπτου από την αρχή Unix time. Ως χρόνος Unix ορίζεται επί του παρόντος, ο αριθμός των δευτερολέπτων που έχουν παρέλθει από τις 00:00:00 UTC της Πέμπτης 1 Ιανουαρίου 1970. Η συγκεκριμένη αναπαράσταση του χρόνου είναι ιδιαίτερα χρήσιμη στα υπολογιστικά συστήματα, καθώς επιτρέπει την εύκολη σύγκριση και τον υπολογισμό των χρονικών διαφορών.

- "measures" - Πρόκειται για ένα αντικείμενο που περιέχει διάφορες μετρήσεις που σχετίζονται με το μετρητή περιβαλλοντικών παραμέτρων "bme280_sensor ":
    - "temperature":23.74: Η θερμοκρασία σε βαθμούς κελσίου (°C).
    - "humidity":67.24, Η σχετική υγρασία - relative humidity (%RH).
    - "pressure":977.55: Η ατμοσφαιρική πίεση (hPa).

4.5. Εκτέλεση εφαρμογής node.js

Το αρχείο execute_script.sh που δημιουργήσαμε είναι ένα shell script, το οποίο θα καλεί τα endpoints 'janitza/d21_get' και 'bosch/bme280_get', για να αντλεί δεδομένα από το μετρητή ενέργειας “Janitza D21” και το μετρητή περιβαλλοντικών παραμέτρων “Bosch BME280”.

Στη συνέχεια, με το εργαλείο Cron, έχουμε τη δυνατότητα στο Ubuntu και γενικά σε Linux συντάγματα, να προγραμματίζουμε την εκτέλεση scripts σε καθορισμένα χρονικά διαστήματα. Οι εργασίες που προγραμματίζονται με αυτόν τον τρόπο ονομάζονται "cron jobs". Με τη χρήση του συγκεκριμένου εργαλείου θα καλούμε το αρχείο execute_script.sh που δημιουργήσαμε προηγουμένως, για να εκτελείται αυτόματα κάθε 15 λεπτά.

Συνεπώς, κάθε 15 λεπτά θα αντλούνται τα δεδομένα με αυτόματο τρόπο, θα μεταφέρονται προσωρινά στις ουρές του RabbitMQ και στη συνέχεια θα καταλήγουν στο Thingsboard, για μόνιμη αποθήκευση και απεικόνιση.

execute_script.sh

```
#execute_script.sh
#API GET
curl --location --request GET 'http://localhost:3000/janitza/d21_get'
curl --location --request GET 'http://localhost:3000/bosch/bme280_get'
```

Εικόνα 73 Κλήση των endpoints 'janitza/d21_get' & 'bosch/bme280_get'

Execute execute_script.sh every 15 minutes

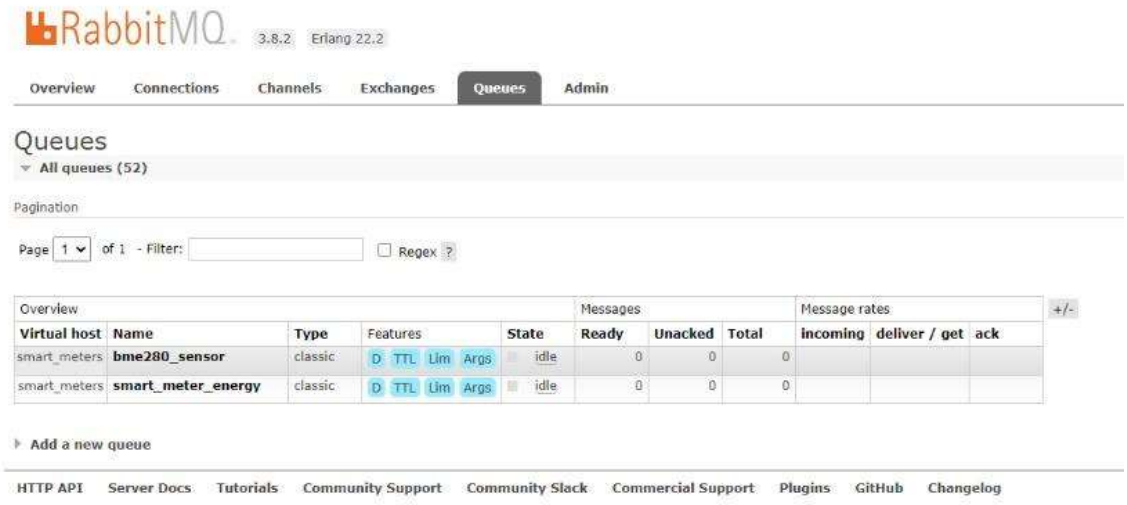
```
*/15 * * * * /home/chryso/execute_script.sh
```

Εικόνα 74 Εκτέλεση του execute_script.sh κάθε 15 λεπτά

4.6. Μεταφορά μετρήσεων από την εφαρμογή στο RabbitMQ

4.6.1. Οι ουρές (queues) του RabbitMQ

Στην παρακάτω οθόνη έχουν δημιουργηθεί δύο ουρές (queues) με όνομα bme280_sensor και smart_meter_energy στις οποίες μεταφέρονται προσωρινά τα δεδομένα από τον αισθητήρα περιβαλλοντικών παραμέτρων BME280 και από το μετρητή ενέργειας Janitza D21, αντίστοιχα. Στις συγκεκριμένες ουρές τα δεδομένα μεταφέρονται προσωρινά πριν μεταφερθούν στο Thingsboard μέσω του πρωτοκόλλου MQTT, για μόνιμη αποθήκευση και απεικόνιση των δεδομένων, στον τελικό χρήστη.

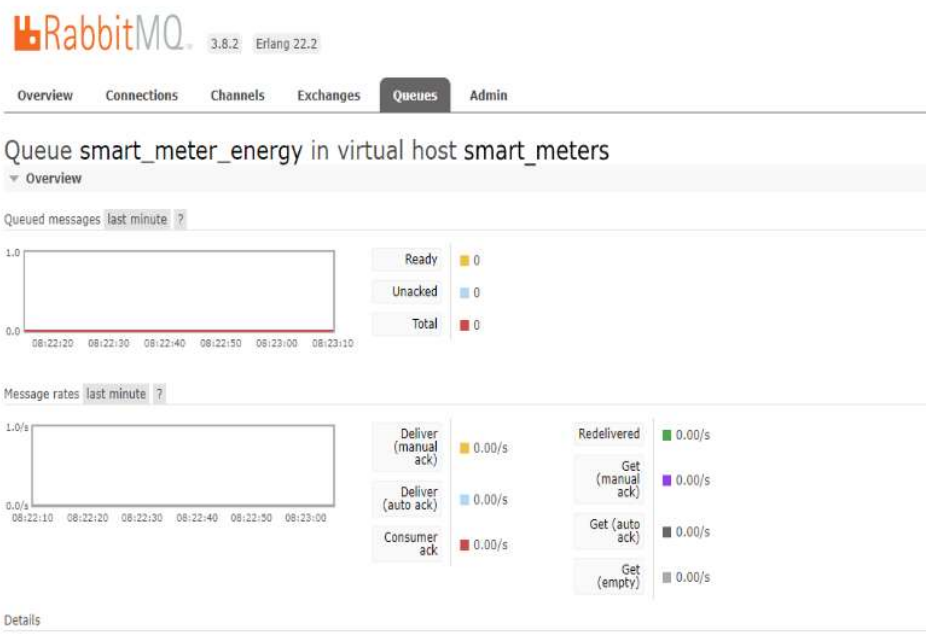


Εικόνα 75 Οι ουρές (queues) του RabbitMQ

4.6.2. Μεταφορά μετρήσεων για το μετρητή ενέργειας Janitza D21

Καλώντας το endpoint της εφαρμογής που δοκιμάσαμε στην ενότητα 4.2.5, αντλεί ο κώδικας που δημιουργήσαμε σε node.js, τις μετρήσεις από το μετρητή ενέργειας "Janitza D21" και τις στέλνει στην ουρά "Smart_meter_energy", επειδή έχουμε θέσει ως κλειδί δρομολόγησης (routing key) στον κώδικα στην ενότητα 4.2.4.3., που φαίνεται παρακάτω το SmartEnergyMeterRoutingKey: **'meter.master'.** Για να γίνει η σύνδεση μεταξύ του κώδικα και του RabbitMQ, θα πρέπει να ορίσουμε το ίδιο κλειδί δρομολόγησης (routing key) στο section: Bindings στην ουρά "Smart_meter_energy" του RabbitMQ.

```
const RABBIT_PARAMS = {
   HOST: config.RABBIT.HOST,
   PORT: config.RABBIT.PORT,
   USER: config.RABBIT.SMART_USER,
   PASSWORD: config.RABBIT.SMART_PASSWORD,
    ExchangeName: 'amq.topic',
    ExchnageType: 'topic',
    SmartEnergyMeterRoutingKey: 'meter.master',
    IsDurable: true
```

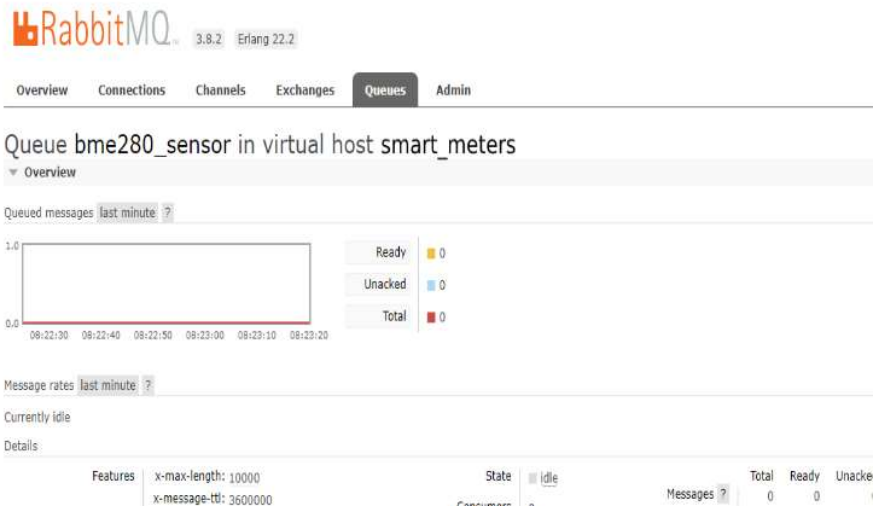


Εικόνα 76 Μεταφορά μετρήσεων για το μετρητή ενέργειας Janitza D21 στο RabbitMQ

4.6.3. Μεταφορά μετρήσεων για το μετρητή περιβαλλοντικών παραμέτρων BME280

Με παρόμοιό τρόπο, όπως και στην προηγούμενη ενότητα, θα πρέπει να εργαστούμε για να συνδέσουμε τον κώδικα του node.js με το RabbitMQ. Καλώντας το endpoint της εφαρμογής που δοκιμάσαμε στην ενότητα 4.4, αντλεί ο κώδικας τις μετρήσεις από το μετρητή περιβάλλοντος “Bosch BME280” και τις στέλνει στην ουρά “bme280_sensor”, επειδή έχουμε θέσει ως κλειδί δρομολόγησης (routing key) το Bme280SensorRoutingKey: 'bme280.master'. Για να γίνει η σύνδεση μεταξύ του κώδικα και του RabbitMQ, θα πρέπει να ορίσουμε το ίδιο κλειδί δρομολόγησης (routing key) στο section: Bindings στην ουρά “ bme280_sensor ” του RabbitMQ.

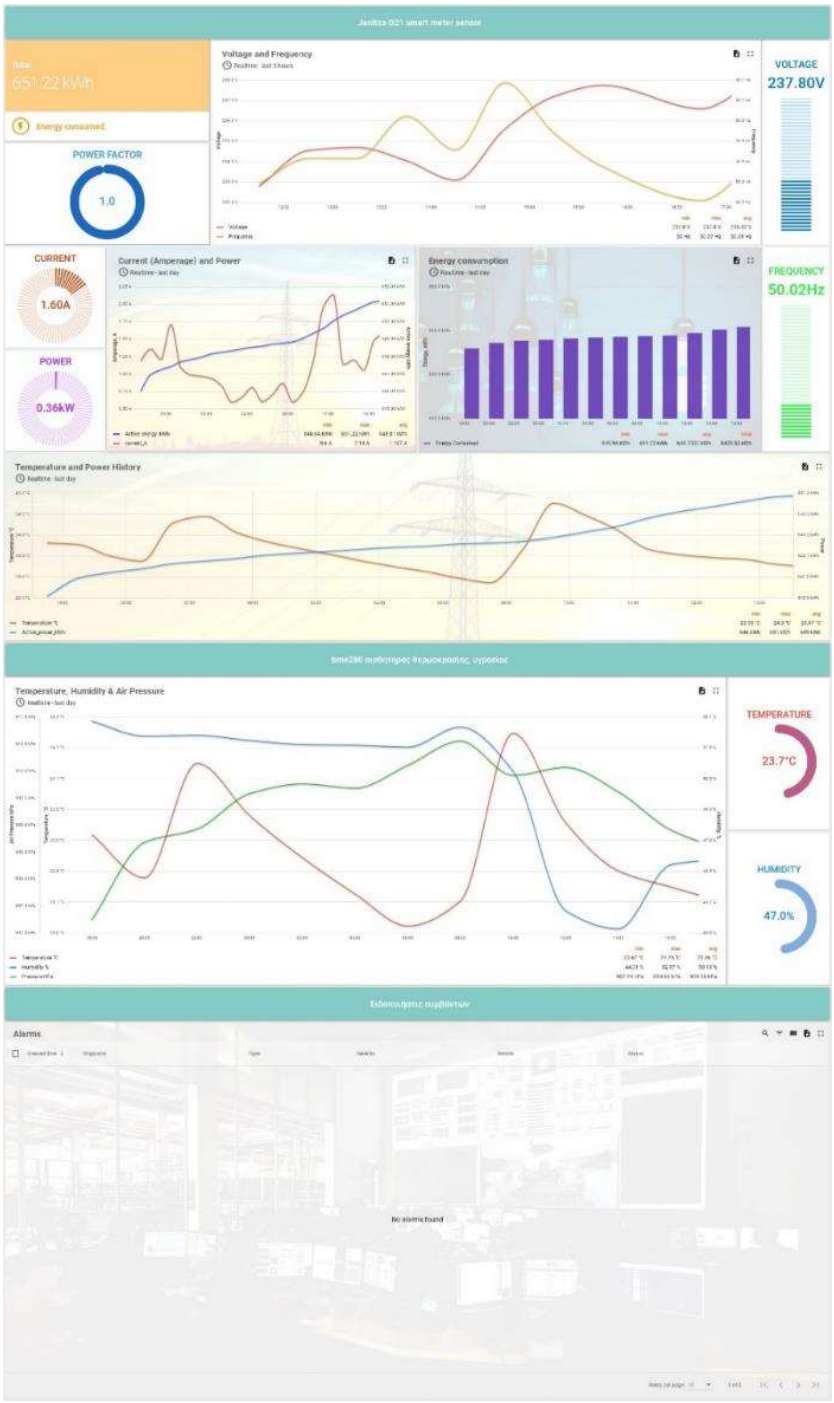
```
const RABBIT_PARAMS = {
    HOST: config.RABBIT.HOST,
    PORT: config.RABBIT.PORT,
    USER: config.RABBIT.SMART_USER,
    PASSWORD: config.RABBIT.SMART_PASSWORD,
    ExchangeName: 'amq.topic',
    ExchnageType: 'topic',
    Bme280SensorRoutingKey: 'bme280.master',
    IsDurable: true
```



Εικόνα 77 Μεταφορά μετρήσεων για το μετρητή περιβάλλοντος BME280 στο RabbitMQ

4.7. Απεικόνιση μετρήσεων στο Thingsboard

Όπως αναφέραμε στα προηγούμενα κεφάλαια, η εφαρμογή node.js διαβάζει τα δεδομένα από τους αισθητήρες και τα προωθεί στο message broker RabbitMQ, για προσωρινή αποθήκευση. Σε αυτό το σημείο, το Thingsboard έχει συνδεθεί με το πρωτόκολλο MQTT στην αντίστοιχη ουρά και μόλις αντιληφθεί δεδομένα, τα αντλεί και τα μεταβιβάζει για μόνιμη αποθήκευση στη βάση δεδομένων PostgreSQL. Στη συνέχεια, τα δεδομένα ανακτώνται από τη βάση και απεικονίζονται με τη χρήση Dashboards, στον τελικό χρήστη. Τα Dashboards μπορούν να προσαρμοστούν για να παρουσιάσουν την πληροφορία στο χρήστη, με τρόπο σαφή και περιεκτικό τόσο για προγενέστερα δεδομένα όσο και για δεδομένα που εισέρχονται στην πλατφόρμα σε πραγματικό χρόνο. Στην παραπάνω διπλανή εικόνα, απεικονίζεται το dashboard που δημιουργήθηκε



Εικόνα 78 Απεικόνιση μετρήσεων στο Thingsboard

για την προβολή και σύγκριση των μετρήσεων που συλλέχθηκαν από τους αισθητήρες χρησιμοποιώντας την εφαρμογή node.js.

Πιο συγκεκριμένα, το dashboard αποτελείται από 3 τμήματα:

- **Μετρήσεις ενέργειας**: Στο πρώτο τμήμα απεικονίζονται οι μετρήσεις από τον αισθητήρα ενέργειας “Janitza D21 Energy Meter”.
- **Μετρήσεις περιβάλλοντος:** Στο δεύτερο τμήμα απεικονίζονται οι μετρήσεις από τον αισθητήρα περιβάλλοντος “Bosch BME280”.
- **Ειδοποιήσεις συμβάντων:** Στο τρίτο τμήμα απεικονίζονται οι ειδοποιήσεις συμβάντων που ενεργοποιούνται όταν εκτελείται ένα γεγονός που έχει οριστεί από τον χρήστη.

4.7.1. Απεικόνιση μετρήσεων για το μετρητή ενέργειας “Janitza D21 Energy Meter”

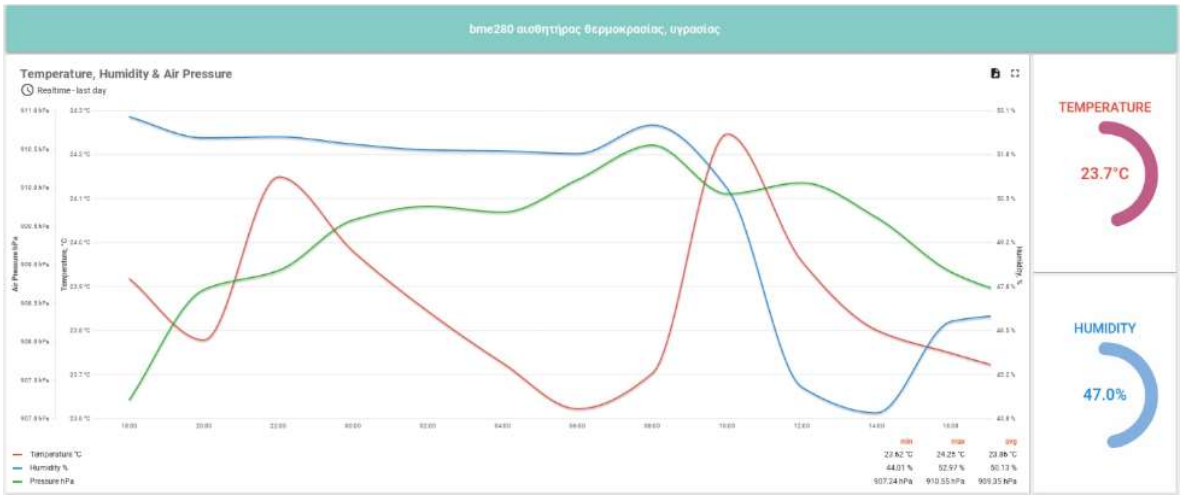
Στο συγκεκριμένο τμήμα εμφανίζονται τα δεδομένα που λαμβάνονται σε πραγματικό χρόνο για τις μετρήσεις: Συντελεστής ισχύος (Power factor), Ρεύμα (Current), Ενεργός ισχύς (Active power), Τάση (Voltage) και Συχνότητα (Frequency). Στα γραφήματα γραμμών και στο γράφημα μπάρας, απεικονίζονται οι μεταβολές για δύο ή περισσότερες μετρήσεις, τόσο σε παρελθοντικό όσο και σε πραγματικό χρόνο. Ο χρήστης μπορεί να επιλέξει και να συγκρίνει τις μετρήσεις που επιθυμεί με βάση τις προηγούμενες ημέρες, εβδομάδες ή μήνες. Συνεπώς, αποτελεί ένα σημαντικό εργαλείο για τον χρήστη, επειδή μπορεί να αντλήσει άμεσα την κατάλληλη πληροφορία έτσι ώστε να επιτύχει όσο γίνεται μεγαλύτερη εξοικονόμηση ενέργειας.

Εικόνα 79 Απεικόνιση μετρήσεων για το μετρητή ενέργειας Janitza D21

4.7.2. Απεικόνιση μετρήσεων για το μετρητή περιβάλλοντος “Bosch BME280 sensor”

Στο συγκεκριμένο τμήμα εμφανίζονται τα δεδομένα που λαμβάνονται σε πραγματικό χρόνο για τις μετρήσεις: θερμοκρασία (temperature) και υγρασία (humidity).

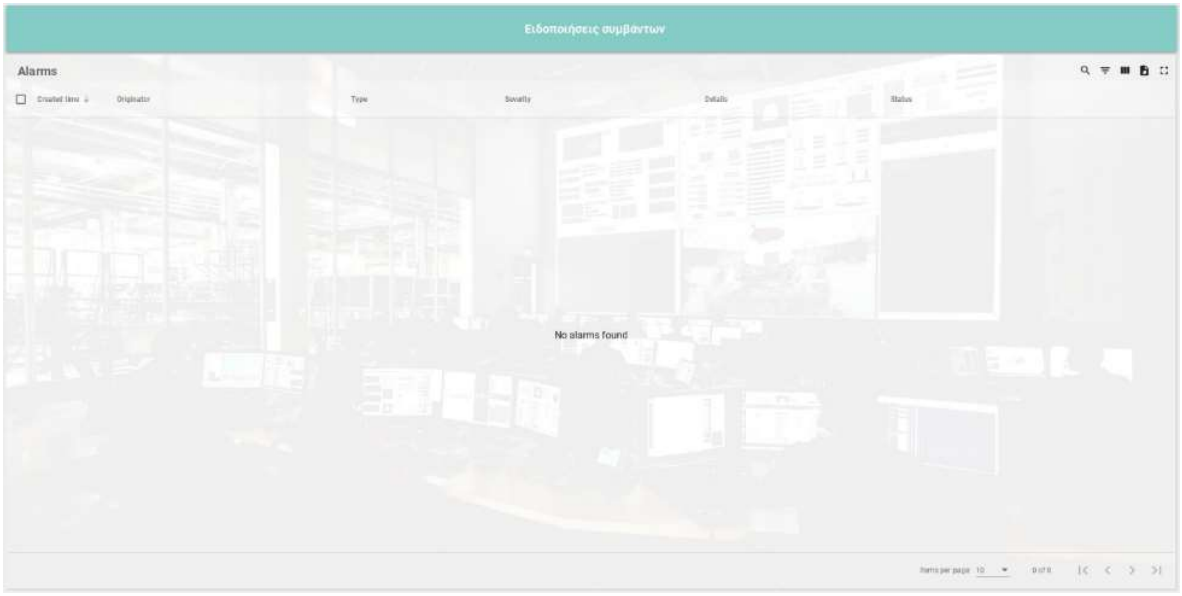
Στα γράφημα γραμμής, απεικονίζονται οι μετρήσεις θερμοκρασία (temperature), υγρασία (humidity) και ατμοσφαιρική πίεση (air pressure) για ένα συγκεκριμένο χρονικό διάστημα. Ο χρήστης μπορεί να τροποποιήσει το συγκεκριμένο χρονικό διάστημα ή να προσθέσει επιπλέον μετρήσεις, με βάση τις ανάγκες του. Συνεπώς, είναι εμφανές ότι είναι πλήρως παραμετροποιήσιμη η αλλαγή ή δημιουργία ενός dashboard από απλούς χρήστες, χωρίς να διαθέτουν τη γνώση συγγραφής κώδικα.



Εικόνα 80 Απεικόνιση μετρήσεων για το μετρητή περιβάλλοντος Bosch BME280

4.7.3. Ειδοποιήσεις Συμβάντων

Το Thingsboard παρέχει δυνατότητα ειδοποίησης για γεγονότα, σύμφωνα με τους κανόνες που έχει θέσει ο χρήστης. Οι ειδοποιήσεις συμβάντων επιτρέπουν στους χρήστες να λαμβάνουν ειδοποιήσεις, όταν συμβαίνουν συγκεκριμένα γεγονότα. Τα γεγονότα που μπορεί να δημιουργηθούν είναι από ένα απλό σενάριο έως πολύπλοκες διαδικασίες. Για παράδειγμα, θα μπορούσε να δημιουργηθεί ένας κανόνας που να ενεργοποιείται όταν η θερμοκρασία είναι μεγαλύτερη από 25°C και η ένταση του ρεύματος ξεπεράσει τα 10A, στον αισθητήρα μέτρησης περιβάλλοντος και στον αισθητήρα μέτρησης ενέργειας, αντίστοιχα, και να ειδοποιείται ο χρήστης μέσω email ή μέσω SMS, ότι δημιουργήθηκε ένα κρίσιμο σφάλμα (critical error).



Εικόνα 81 Ειδοποιήσεις συμβάντων στο Thingsboard

Συμπεράσματα και προτάσεις επέκτασης της εργασίας

Η παρούσα διπλωματική εργασία επικεντρώθηκε στην ανάλυση και ανάγνωση των αισθητήρων μέτρησης ενέργειας “Janitza D21” και μέτρησης περιβάλλοντος “Bosch BME280”, αντίστοιχα. Για τους συγκεκριμένους αισθητήρες δημιουργήθηκε εφαρμογή API σε node.js, έτσι ώστε να αντλούνται από τους καταχωρητές (registers) οι επιθυμητές μετρήσεις, με σκοπό να μεταφέρονται για μόνιμη αποθήκευση και απεικόνιση από το λογισμικό Thingsboard. Για τη σύνδεση των αισθητήρων χρησιμοποιήθηκε το Raspberry Pi. Εκτός της σύνδεσης τον αισθητήρων, το Raspberry Pi χρησιμοποιήθηκε σαν server για να εκτελεί την εφαρμογή και να αντλεί τα δεδομένα των αισθητήρων.

Με τη συνδεσμολογία που απεικονίζεται στην εικόνα 56, αντλήθηκαν δεδομένα από τους αισθητήρες για σύνολο 15 ημερών. Η ενημέρωση των μετρήσεων πραγματοποιείται σε πραγματικό χρόνο και απεικονίζονται στο dashboard του Thingsboard με εποπτικό τρόπο. Με αυτό τον τρόπο έχουμε άμεσα διαθέσιμη την πληροφορία, για την καθημερινή κατανάλωση ενέργειας. Για να πετύχουμε περισσότερη εξοικονόμηση ενέργειας, δημιουργήθηκαν δύο κανόνες (triggers) οι οποίοι θα ενεργοποιούνται και θα αποστέλλουν ειδοποιήσεις (μέσω e-mail), όταν στην πρώτη περίπτωση η ενεργός ενέργεια (Active energy - KWh) υπερβεί ένα συγκεκριμένο όριο (threshold), ενώ στην δεύτερη, όταν ο συντελεστής ισχύος (Power factor - (cos)) υποχωρήσει κάτω από ένα συγκεκριμένο όριο. Συνεπώς, με τη χρήση των παραπάνω εργαλείων, μας παρέχονται πληθώρα επιλογών για να κατασκευάσουμε τους κανόνες που επιθυμούμε με βάση τις ανάγκες μας για να πετύχουμε εξοικονόμηση ενέργειας και μείωση κόστους.

Μία μελλοντική επέκταση τις παρούσας διπλωματικής εργασίας, θα μπορούσε να είναι η τεκμηρίωση (documentation) και η δοκιμή (testing) του συγκεκριμένου κώδικα που υπάρχει στο παράρτημα της διπλωματικής, έτσι ώστε να βεβαιωθούμε ότι η εφαρμογή API που υλοποιήθηκε, λειτουργεί χωρίς σφάλματα. Επιπλέον, ένα ακόμη σενάριο επέκτασης της εργασίας θα μπορούσε να είναι η προσθήκη επιπλέον μετρητών – αισθητήρων, με τη μελέτη λειτουργίας των αντίστοιχων datasheets. Για παράδειγμα, θα μπορούσε να προστεθεί η δυνατότητα να λαμβάνουμε δεδομένα για την καταγραφή της κατανάλωσης του νερού σε μία παροχή που επιθυμούμε να καταγράψουμε. Συνεπώς, στόχος αποτελεί η επαύξηση της λειτουργικότητας της εφαρμογής με πληθώρα αισθητήρων χαμηλού κόστους, που υπάρχουν διαθέσιμοι στο εμπόριο. Με αυτό τον τρόπο θα πραγματοποιείται η ενημέρωση των χρηστών

σε πραγματικό χρόνο, ειδικά για τα κρίσιμα γεγονότα (alerts) που καταγράφονται από τους αισθητήρες και αποτελούν ωφέλιμη πληροφορία για τους χρήστες. Αυτό που θέλουμε να τονίσουμε είναι το εξής σενάριο. Έστω, ότι έχουμε εγκαταστήσει ένα αισθητήρα παρουσίας καπνού και θέλουμε να ενημερωθούμε σε πραγματικό χρόνο (μέσω SMS ή email), για την παρουσία καπνού σε ένα χώρο που θέλουμε να ελέγξουμε. Υλοποιώντας το παραπάνω σενάριο, οι πληροφορίες θα μεταβιβάζεται άμεσα στους χρήστες έτσι ώστε να προβαίνουν άμεσα στις κατάλληλες ενέργειες.

Βιβλιογραφία

[1] Rose, K., Eldridge, S. και Chapin, L., «The Internet of Things: AN OVERVIEW Understanding the Issues and Challenges of a More Connected World',» October 2015.

[2] Mehmood, Y., Ahmad, F, Yaqood, I, Adnane, A., Muhammed, I. και Guizani, S., «Internet-of-Things-Based Smart Cities: Recent Advances and Challenges,» *IEEE Communications Magazine,* τόμ. 55, αρ. 9, pp. 16-24, September 2017.

[3] Alsarysah, O, Mashal, I και Chung, T. Y., «Energy-aware services composition for Internet of Things,» *IEEE 4th World Forum on Internet of Things (WF-IoT),* 05-08 February 2018.

[4] Bedhief, I., Kassar, M. και Aguili, T., «SDN-based architecture challenging the IoT heterogeneity,» *IEEE 3rd Smart Cloud Networks & Systems (SCNS), Dubai, United Arab Emirates,* December 2016.

[5] Vaigandla, K. K., «Communication Technologies and Challenges on 6G Networks for the Internet: Internet of Things (IoT) Based Analysis,» *IEEE International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India,* 23-25 February 2022.

[6] Menniti, D., Pinnarelli, A., Sorrentino, N., Vizza, P., Brusco, G., Polizzi, G., Escalante, J. J. P. και Ayci, D., «IoT Devices and Smart Appliances for Flexibility Services in Unical Campus,» *IEEE Workshop on Blockchain for Renewables Integration (BLORIN), Palermo, Italy,* 02-03 September 2022.

[7] Osako, F. L., Matsubayashi, O. M., Takey, M. S., Cauchick-Miguel, A. P. και Zancul, E., «Cost evaluation challenges for Internet of Things (IoT) based Product/Service-Systems (PSS),» *Elsevier, Procedia CIRP 84,,* pp. 302-306, 2019.

[8] Singh, S. και Baranwal, G., «Quality of Services (QoS) in Internet of Things,» *IEEE International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, India,* 23-24 February 2018.

[9] Iqbal, W., Abbas, H., Daneshmand, M., Rauf, B. και Bangash, A. Y., «An In-Depth Analysis of IoT Security Requirements Challenges, and Their Countermeasures via Software-Defined Security,» *IEEE Internet of Things Journal,* τόμ. 7, October 2020.

[10] Kang, K.-D., «A Review of Efficient Real-Time Decision Making in the Internet of Things,» *MDPI technologies,* τόμ. 10, January 2022.

[11] Abubakari, S, «Improving Business Effectiveness Using Internet of Things,» *Diverse Journal of Computer and Information Sciences (DJCIS,* τόμ. 1, December 2019.

[12] Stolojescu-Crisan, C, Crisan, C. και Butunoi, Bogdan-Petru, «An IoT-Based Smart Home Automation System,» *MDPI Sensors,* τόμ. 21, May 2021.

[13] Borisovskaya, A. V. και Turlikov, A. M., «Reducing Energy Consumption in the IoT Systems with Unlimited Number of Devices,» *IEEE 2021 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF), St. Petersburg, Russia,* 31 May 2021-04 June 2021.

[14] Ho-Sam-Sooi, N., Pieters, W. και Kroesen, M., «Investigating the effect of security and privacy on IoT device purchase behavior,» *Computers & Security, Elsevier,* τόμ. 12, March 2021.

[15] Bonetto, R, Bui, N., Lakkundi, V., Olivereau, A., Serbanati, A. και Rossi, M., «Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples,» *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), San Francisco, USA,* June 2012.

[16] Tejveer Anand, Sourabh Upare, Siddhant Jain, Maithili Andhare και Kishor Bhangale, «Deployment of Real-Time Energy Monitoring System Using IoT,» *3rd International Conference for Emerging Technology (INCET),* 27-29 May 2022.

[17] Eyhab Al-Masri, Karan Raj Kalyanam, John Batts, Jonathan Kim, Sharanjit Singh, Tammy Vo και Charlotte Yan, «Investigating Messaging Protocols for the Internet of Things (IoT),» *IEEE Access,* τόμ. 8, pp. 94880 - 94911, 08 May 2020.

[18] Cavada, M., Hunt, D. V. L. και Rogers, C. D. F., «Smart Cities: Contradicting Definitions

and Unclear Measures,» *4th World Sustainability Forum,* November 2014.

[19] Kumar, T. M. V. και Dahiya, B., «Smart Economy and Smart Cities,» *Springer,* pp. 3-76.

[20] Nastjuk, I., Trang, S. και Papageorgiou, E. I., «Smart Cities and smart governance models for future cities,» *Electronic Markets, Springe,* τόμ. 32, pp. 1917-1924, November 2022.

[21] Gupta, S., Mustafa, S. Z. και Kumar, H., «3: Smart People for Smart Cities: A Behavioral Framework for Personality and Roles: Smarter People, Governance, and Solutions,» *Advances In Smart Cities,* pp. 23-30, May 2017.

[22] Arce-Ruiz, R. M., Baucells, N. και Alonso, C. M., «Smart Mobility and Smart Cities,» *CIT2016. XII Congreso de Ingenieria del Transporte, Valencia, Spain,* pp. 1209-1219.

[23] Chang, S. και Smith, M. K., «Residents' Quality of Life in Smart Cities: A Systematic Literature Review,» *MDPI land,* April 2023.

[24] Liu, L. και Zhang, Y., «Smart environmental design planning for smart city based on deep learning,» *Sustainable Energy Technologies and Assessments, Elsevier,* τόμ. 47, October 2021.

[25] Saloni, S. και Hegde, A., «WiFi-aware as a connectivity solution for IoT pairing IoT with WiFi aware technology: Enabling new proximity based services,» *IEEE International Conference on Internet of Things and Applications (IOTA), Pune, India,* 22-26 January 2016.

[26] Sofi, A. M., «Bluetooth Protocol in Internet of Things (IoT), Security Challenges and a Comparison with Wi-Fi Protocol: A Review',» *International Journal of Engineering and Technical Research,* τόμ. 5, αρ. 11, November 2016.

[27] Gavra, V.-D. και Pop, A. O., «Usage of Zigbee and LoRa wireless technologies in IoT systems,» *IEEE International Symposium for Design and Technology in Electronic Packaging, Pitesti, Romania,* 21-24 October 2020.

[28] A. Zanella, N. Bui, A. Castellani, L. Vangelista και M. Zorze, «Internet of Things for

Smart Cities,» *IEEE Internet of Things Journal,* τόμ. 1.1, pp. 22-32, 2014.

[29] Mishra, B. και Kertesz, A., «The Use of MQTT in M2M and IoT Systems: A Survey,» *IEEE Access,* τόμ. 8, pp. 201071-201086, November 2020.

[30] Ansari, B. D., Rehman, A.-U. και Ali, R., «Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP,» *International Journal of Computer Applications,* τόμ. 179, pp. 9-14, March 2018.

[31] Fernandes, L. J., Lopes, C. I., Rodrigues, C. P. J. J. και Ullah, S., «Performance evaluation of RESTful web services and AMQP Protocol,» *Fifth International Conference on Ubiquitous and Future Networks (ICUFN), Da Nang, Vietnam,* 02-05 July 2013.

[32] You, W. και Ge, H., «Design and Implementation of Modbus Protocol for Intelligent Building Security,» *IEEE 19th International Conference on Communication Technology (ICCT), Xian, China,* 16-19 October 2019.

[33] «Official website for Node.js,» [Ηλεκτρονικό]. Available: https://nodejs.org/. [Πρόσβαση 26 06 2023].

[34] «Official website for RabbitMQ,» [Ηλεκτρονικό]. Available: https://www.rabbitmq.com. [Πρόσβαση 26 06 2023].

[35] «The analysis of the performance of RabbitMQ and ActiveMQ,» *14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER),* 24-26 September 2015.

[36] «Official website for ThingsBoard,» [Ηλεκτρονικό]. Available: https://thingsboard.io. [Πρόσβαση 26 06 2023].

[37] Janitza Electronics, «DIN Rail Mounted Energy Meter - D21 | User manual and technical data,» 10 2018. [Ηλεκτρονικό]. Available: https://www.janitza.com/.

[38] «Official website for Bosch BME280 sensor,» [Ηλεκτρονικό]. Available: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-

bme280/. [Πρόσβαση 26 06 2023].

[39] Bosch, «BME280 – Data sheet,» 01 2022. [Ηλεκτρονικό]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf. [Πρόσβαση 26 06 2022].

[40] «Official site for Postman,» [Ηλεκτρονικό]. Available: https://www.postman.com/. [Πρόσβαση 26 06 2023].

[41] Muneer Bani Yassein, Mohammed Q. Shatnawi, Shadi Aljwarneh και Razan Al-Hatmi, «Internet of Things: Survey and open issues of MQTT protocol,» *International Conference on Engineering & MIS (ICEMIS),* 08-10 May 2017.

[42] Muneer Bani Yassein, Mohammed Q. Shatnawi, Shadi Aljwarneh και Razan Al-Hatmi, «Internet of Things: Survey and open issues of MQTT protocol,» *International Conference on Engineering & MIS (ICEMIS),* 08-10 May 2017.

Παράρτημα

Κώδικας για το μετρητή ενέργειας “Janitza D21 Energy Meter”

```
var express = require('express');
var router = express.Router();

const winstonLogger = require('../src/winstonLogger');

const config = require('../src/config');
const rabbit = require('../src/rabbit');

const ModbusRTU = require("modbus-serial");
const client = new ModbusRTU();

client.connectRTUBuffered('/dev/ttyUSB0', { baudRate: 9600 });
client.setID(2);

const mom_tz = require('moment-timezone');

const RABBIT_PARAMS = {
    HOST: config.RABBIT.HOST,
    PORT: config.RABBIT.PORT,
    USER: config.RABBIT.BIN_USER,
    PASSWORD: config.RABBIT.BIN_PASSWORD,
    ExchangeName: 'amq.topic',
    ExchnageType: 'topic',
    SmartEnergyMeterRoutingKey: 'meter.master',
    IsDurable: true
}

router.get('/d21_get', async function (req, res, next) {
    try {
        winstonLogger.log('debug', req.query);
        const virtual_host = 'smart_meters';
        const channel = await rabbit.GetChannel(virtual_host, RABBIT_PARAMS);

        if (!channel) {
            res.send("No Rabbit channel available !");
            return;
        }

        let measure = await GetDataSmartMeterHome(client, 0, 22);
                            rabbit.SendToRabbitmq(channel,      RABBIT_PARAMS.ExchangeName,
RABBIT_PARAMS.SmartEnergyMeterRoutingKey, measure);
```

```
        res.send(measure);

    } catch (e) {
        winstonLogger.error(e);
        res.send(e);
    }
});


async function GetDataSmartMeterHome(client, reg_start, reg_finish){
    try {
        const now = mom_tz.utc().tz("Europe/Athens").format();
        const timestampMs = Date.parse(now);
        let measure = await client.readHoldingRegisters(reg_start, reg_finish);
        let voltage_V =parseFloat(measure.buffer.readFloatBE(0).toFixed(2));
        let current_A = parseFloat(measure.buffer.readFloatBE(4).toFixed(2));
        let Active_power_kW = parseFloat(measure.buffer.readFloatBE(8).toFixed(2));
        let Reactive_power_kvar = parseFloat(measure.buffer.readFloatBE(12).toFixed(2));
        let Apparent_power_kVA = parseFloat(measure.buffer.readFloatBE(16).toFixed(2));
        let Power_factor = parseFloat(measure.buffer.readFloatBE(20).toFixed(2));
        let frequency_Hz = parseFloat(measure.buffer.readFloatBE(24).toFixed(2));

        const data = {
            devid: "Janitza_D21_Energy_Meter",
            timestamp: now,
            timestampMs: timestampMs,
            measures:  {
                "voltage_V": voltage_V,
                "current_A": current_A,
                "Active_power_kW": Active_power_kW,
                "Reactive_power_kvar": Reactive_power_kvar,
                "Apparent_power_kVA": Apparent_power_kVA,
                "Power_factor": Power_factor,
                "frequency_Hz": frequency_Hz
            }
        }

        return data;

    } catch (e) {
        winstonLogger.error(e);
        return [];
    }
}

module.exports = router;
```

Κώδικας για το μετρητή περιβάλλοντος “Bosch BME280 sensor”

```
var express = require('express');
var router = express.Router();

const winstonLogger = require('../src/winstonLogger');

const config = require('../src/config');
const rabbit = require('../src/rabbit');

const i2c = require('i2c-bus');

const port = 1;
const address = 0x76;
const bus = i2c.openSync(port);

const mom_tz = require('moment-timezone');

// Defining oversampling mode constants
const OVERSAMPLING_X1 = 1;

const RABBIT_PARAMS = {
  HOST: config.RABBIT.HOST,
  PORT: config.RABBIT.PORT,
  USER: config.RABBIT.CITY_HALL_USER ,
  PASSWORD: config.RABBIT.CITY_HALL_PASSWORD ,
  ExchangeName: 'amq.topic',
  ExchnageType: 'topic',
  Bme280SensorRoutingKey: 'bme280.master',
  IsDurable: true
}

router.get('/bme280_get', async function (req, res, next) {
    try {
        winstonLogger.log('debug', req.query);
        const virtual_host = 'smart_meters';
        const channel = await rabbit.GetChannel(virtual_host, RABBIT_PARAMS);

        const now = mom_tz.utc().tz("Europe/Athens").format();
        const timestampMs = Date.parse(now);

        if (!channel) {
          res.send("No Rabbit channel available !");
          return;
      }

      const initParameters = await bme280Init();
```

```
      const data = await bme280Start(initParameters);

        const json =  {
              devid: "bme280_sensor",
              timestamp: now,
              timestampMs: timestampMs,
              measures:  {
                  "temperature": data.temperature.toFixed(2),
                  "humidity": data.humidity.toFixed(2),
                  "pressure": data.pressure.toFixed(2)
              }
            };

                rabbit.SendToRabbitmq(channel,     RABBIT_PARAMS.ExchangeName,
RABBIT_PARAMS.Bme280SensorRoutingKey, json);

      res.send(json);

    } catch (e) {
      winstonLogger.error(e);
      res.send(e);
    }
  });


  async function bme280Init() {
    const calib = {};

    // Temperature trimming params
    calib['dig_T1'] = unsignedShort(0x88);
    calib['dig_T2'] = signedShort(0x8A);
    calib['dig_T3'] = signedShort(0x8C);

    // Pressure trimming params
    calib['dig_P1'] = unsignedShort(0x8E);
    calib['dig_P2'] = signedShort(0x90);
    calib['dig_P3'] = signedShort(0x92);
    calib['dig_P4'] = signedShort(0x94);
    calib['dig_P5'] = signedShort(0x96);
    calib['dig_P6'] = signedShort(0x98);
    calib['dig_P7'] = signedShort(0x9A);
    calib['dig_P8'] = signedShort(0x9C);
    calib['dig_P9'] = signedShort(0x9E);

    // Humidity trimming params
    calib['dig_H1'] = unsignedByte(0xA1);
    calib['dig_H2'] = signedShort(0xE1);
```

```
    calib['dig_H3'] = unsignedByte(0xE3);
    calib['dig_H4'] = (signedByte(0xE4) << 4) | (signedByte(0xE5) & 0x0F);
    calib['dig_H5'] = ((signedByte(0xE5) >> 4) & 0x0F) | (signedByte(0xE6) <<
4);
    calib['dig_H6'] = signedByte(0xE7);

    return calib;
  }

  function rawMeasurements(block) {
    const pressure = (block[0] << 16 | block[1] << 8 | block[2]) >> 4;
    const temperature = (block[3] << 16 | block[4] << 8 | block[5]) >> 4;
    const humidity = block[6] << 8 | block[7];

    return {
      temperature: temperature,
      pressure: pressure,
      humidity: humidity
    };
  }

  function calcMeasurements(rawReadings, compensationParams) {

    let temperature = 0.0, humidity = 0.0, pressure = 0.0;

    // Calculate temperature
    const v1 = (rawReadings.temperature / 16384.0 - compensationParams.dig_T1
/ 1024.0) * compensationParams.dig_T2;
        const   v2   =   (Math.pow(rawReadings.temperature   /   131072.0   -
compensationParams.dig_T1 / 8192.0, 2)) * compensationParams.dig_T3;
    temperature = (v1 + v2) / 5120.0;

    // Calculate humidity
    let res = temperature - 76800.0;
      res  =  (rawReadings.humidity  -  (compensationParams.dig_H4  *  64.0  +
compensationParams.dig_H5  /  16384.0  *  res))  *  (compensationParams.dig_H2  /
65536.0  *  (1.0  +  compensationParams.dig_H6  /  67108864.0  *  res  *  (1.0  +
compensationParams.dig_H3 / 67108864.0 * res)));
    res = res * (1.0 - compensationParams.dig_H1 * res / 524288.0);
    humidity = Math.max(0.0, Math.min(res, 100.0));

    // Calculate pressure
    let v1p = temperature / 2.0 - 64000.0;
    let v2p = v1p * v1p * compensationParams.dig_P6 / 32768.0;
    v2p = v2p + v1p * compensationParams.dig_P5 * 2.0;
    v2p = v2p / 4.0 + compensationParams.dig_P4 * 65536.0;
```

```
      v1p = (compensationParams.dig_P3 * v1p * v1p / 524288.0 +
compensationParams.dig_P2 * v1p) / 524288.0;
    v1p = (1.0 + v1p / 32768.0) * compensationParams.dig_P1;

    // Prevent divide by zero
    if (v1p === 0) {
      res = 0;
    } else {
      res = 1048576.0 - rawReadings.pressure;
      res = ((res - v2p / 4096.0) * 6250.0) / v1p;
      v1p = compensationParams.dig_P9 * res * res / 2147483648.0;
      v2p = res * compensationParams.dig_P8 / 32768.0;
      res = res + (v1p + v2p + compensationParams.dig_P7) / 16.0;
      pressure = res / 100.0;
    }

    return {
      temperature: temperature,
      humidity: humidity,
      pressure: pressure,
    };
  }

  async function bme280Start(params) {
    const mode = 1;  // forced
    const tOversampling = OVERSAMPLING_X1;
    const hOversampling = OVERSAMPLING_X1;
    const pOversampling = OVERSAMPLING_X1;

    // ctrl_hum
    bus.writeByteSync(address, 0xF2, hOversampling);
    // ctrl_meas
    bus.writeByteSync(address, 0xF4, tOversampling << 5 | pOversampling << 2 |
mode);
    await sleep(10);

    const buffer = Buffer.alloc(8);
    bus.readI2cBlockSync(address, 0xF7, 8, buffer);
    const raw_data = rawMeasurements(buffer);

    const data_readings = calcMeasurements(raw_data, params);
    return data_readings;
  }


  function unsignedShort(register) {
    return bus.readWordSync(address, register) & 0xffff;
```

```
}

function signedShort(register) {
  const word = unsignedShort(register);
  return word < 0x8000 ? word : word - 0x10000;
}

function unsignedByte(register) {
  return bus.readByteSync(address, register) & 0xff;
}

function signedByte(register) {
  const byte = unsignedByte(register);
  return byte < 0x80 ? byte : byte - 0x100;
}

function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

module.exports = router;
```