ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ IOT ΑΣΥΡΜΑΤΟΥ ΕΛΕΓΧΟΥ ΚΑΙ ΜΕΤΡΗΣΗΣ ΕΝΕΡΓΕΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΠΑΚΑ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων: Μηνάς Δασυγένης

Επίκουρος Καθηγητής

Εργαστήριο Ρομποτικής, Ενσωματωμένων & Ολοκληρωμένων Συστημάτων

ΚΟΖΑΝΗ/ΟΚΤΩΒΡΙΟΣ/2022

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING

ece.uowm.gr

DESIGN AND IMPLEMENTATION OF AN IOT SYSTEM FOR WIRELESS CONTROL AND ENERGY MEASURE

THESIS

BAKAS PANAGIOTIS

SUPERVISOR: Minas Dasygenis

Assistant Professor

Robotics, Embedded and Integrated Systems Laboratory

KOZANI/OCTOBER/2022

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Σχεδιασμός και υλοποίηση συστήματος IoT, ασύρματου ελέγχου και μέτρησης ενέργειας" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Δασυγένη Μηνά αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: Παναγιώτης Μπάκας

Περίληψη

Στην παρούσα διπλωματική εργασία, σχεδιάστηκε και υλοποιήθηκε ένα σύστημα διαδικτύου των πραγμάτων (IoT) με σκοπό τον απομακρυσμένο έλεγχο ηλεκτρικών συσκευών, καθώς και την μέτρηση και διατήρηση ιστορικών δεδομένων για την κατανάλωση των συσκευών αυτών. Στην εισαγωγή της διπλωματικής, γίνεται αναφορά στην ανάπτυξη της τεχνολογίας του διαδικτύου των πραγμάτων, αλλά και το πως μπορεί ο σύγχρονος άνθρωπος να αξιοποιήσει τέτοιες συσκευές στην καθημερινότητα του.

Η εργασία αυτή αρχίζει παρουσιάζοντας τις τεχνολογίες που χρησιμοποιήθηκαν τόσο στο λογισμικό, όσο και στο υλικό της μέρος, και τα εργαλεία που αξιοποιήθηκαν για την ολοκλήρωση της. Στην συνέχεια, ορίζονται οι απαιτήσεις της λειτουργικότητας για το σύστημα που υλοποιήθηκε. Τα βασικότερα διακριτά τμήματα για τον χρήστη είναι μια συσκευή με συνδεσιμότητα WiFi και Bluetooth, υπεύθυνη για τον έλεγχο μιας πρίζας και την μέτρηση της κατανάλωσης, αλλά και μια κινητή εφαρμογή για την απομακρυσμένη πρόσβαση του χρήστη μέσα από το διαδίκτυο. Το σύστημα χαρακτηρίζεται από απλότητα και ευχρηστία. Τα κυριότερα κομμάτια της έρευνας αφορούν την χρήση πολλαπλών πρωτοκόλλων για την επικοινωνία και μεταφορά των δεδομένων ανάμεσα σε συσκευές διαφορετικής φύσεως, την αρμονική συνεργασία των διάφορων υποσυστημάτων υλικού και λογισμικού αλλά και, την διαχείριση του μεγάλου όγκου δεδομένων που παράγονται από τις συσκευές αυτές κατά την χρήση τους (π.χ. μέτρηση της κατανάλωσης), με την αξιοποίηση σύγχρονων τεχνολογιών και τεχνικών.

Η συσκευή που κατασκευάστηκε στην εν λόγω εργασία, μπορεί να χρησιμοποιηθεί είτε ως κινητή, είτε να προσαρτηθεί σε κάποιον τοίχο. Ωστόσο, απαραίτητη προϋπόθεση είναι η ύπαρξη μια σταθερής σύνδεσης ασύρματου δικτύου WiFi για την πρόσβαση της στο διαδίκτυο.

Στόχος του υλοποιημένου έργου είναι, να διευκολύνει τον χρήστη του συστήματος στην καθημερινότητα του με την δυνατότητα του απομακρυσμένου ελέγχου που παρέχεται, ενώ παράλληλα να τον καταστήσει πιο υπεύθυνο σε θέματα κατανάλωσης ενέργειας μέσω της παρακολούθησης της κατανάλωσης των συσκευών του, βοηθώντας τον έτσι να αποκτήσει ενεργειακή συνείδηση.

Λέξεις Κλειδιά: διαδίκτυο των πραγμάτων, μέτρηση κατανάλωσης, απομακρυσμένος έλεγχος, συσκευή, WiFi, τεχνολογία, ενέργεια.

Abstract

In the present diploma thesis, there was designed and implemented an internet of things (IoT) system for remote control of electrical appliances and also storing consumption measurement data for these appliances. In the introduction, the diploma thesis refers to the development of internet of things technology, but also how it can be utilized by contemporary humans the everyday life.

The diploma thesis starts by showcasing the technologies used both in the software and the hardware section, and also the tools that were utilized for its completion. Then, there are defined the functionality requirements of the implemented system, whose most distinct parts for the user are a device with WiFi and Bluetooth connectivity and a mobile app for remote access to the device through the internet. The whole system is characterized by simplicity and ease of use. The main parts of the research are about the usage of many different protocols for the communication and data transfer between different kinds of devices, the smooth interplay of the various hardware and software subsystems, and also the management of a large volume of data that are being produced by the devices (E.g. by measuring the consumption), by utilizing modern technologies and technics.

The device that was made for this diploma thesis can be used as a mobile or it can be attached to a wall. However, a necessary condition is the existence of a stable connection to a wireless WiFi network for it to access the internet.

The goal of the implemented system is to facilitate the user in the everyday life using the remote control capability that the system offers, and also enable the user to be more responsible for energy consumption issues through monitoring of his devices, helping him to acquire energy awareness.

Keywords: internet of things, consumption measurement, remote control, device, WiFi, technology, energy.

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω την οικογένεια μου, για την στήριξη της και για όσα μου πρόσφερε κατά την διάρκεια των σπουδών μου. Επίσης, τους καθηγητές του Πανεπιστημίου Δυτικής Μακεδονίας για τις γνώσεις που μου μετέδωσαν και ιδιαίτερα τον Δρ. Μηνά Δασυγένη για την συνεργασία και την καθοδήγηση του κατά την εκπόνηση αυτής της διπλωματικής εργασίας. Ακόμη, θέλω να ευχαριστήσω τους συνάδελφους και φίλους για τις στιγμές που έζησα μαζί τους τα χρόνια των σπουδών μου και με βοήθησαν να αναπτυχθώ ως μηχανικός αλλά και ως άτομο.

Περιεχόμενα

Περίληψη 7
Abstract 8
Ευχαριστίες 10
Περιεχόμενα 12
Κατάλογος Εικόνων 15
Κατάλογος Πινάκων 17
Κεφάλαιο 1: Εισαγωγή 18
1.1 Σκοπός του έργου 18
1.2 Παρόμοιες εργασίες 19
1.2.1 TP-LINK Tapo P110 19
1.2.2 Amazon Smart Plug 19
1.2.3 Shelly Plug 20
1.3 Συμπεράσματα παρόμοιων συστημάτων 20
1.4 Σύνοψη κεφαλαίου 20
Κεφάλαιο 2: Θεωρητικό Υπόβαθρο 21
2.1 Πρωτόκολλα & τεχνολογίες που χρησιμοποιήθηκαν 21
2.1.1 HTTP & HTTPS 21
2.1.2 MQTT 22
2.1.3 BLE - Bluetooth 23
2.1.4 Modbus RTU 24
2.2 Λογισμικό μέρος 25
2.2.1 Η γλώσσα προγραμματισμού Javascript 25
2.2.2 Το περιβάλλον εκτέλεσης Node.js 26
2.2.3 Το πλαίσιο εφαρμογής Express.js 27
2.2.4 Το πλαίσιο κινητών εφαρμογών React Native 28
2.2.5 Η γλώσσα προγραμματισμού C/C++ 29

2.2.6 Η βάση δεδομένων MongoDB 30
2.3 Εργαλεία 30
2.3.1 Visual Studio Code 30
2.3.2 Arduino IDE 30
2.3.3 Android Studio 31
2.4 Υλικό μέρος 31
2.4.1 Ο Μικροελεγκτής ESP-32 31
2.4.2 Ο μετρητής κατανάλωσης SDM120M 32
2.4.3 Ο μετατροπέας TTL σε RS485 33
2.4.4 Το τροφοδοτικό HI-LINK 33
2.4.5 Το ρελέ 34
2.5 Σύνοψη κεφαλαίου 34
Κεφάλαιο 3: Απαιτήσεις Συστήματος 35
3.1 Απαιτήσεις συστήματος 35
3.2 Περιπτώσεις χρήσης 36
3.2.1 Μη συνδεδεμένος χρήστης 36
3.2.2 Συνδεδεμένος χρήστης 36
3.2.3 Διαχειριστής 38
3.2.4 Συσκευή 39
3.3 Βάση δεδομένων 39
3.3.1 Σχεσιακό διάγραμμα της βάσης δεδομένων 40
3.3.2 Συλλογές της βάσης δεδομένων 40
3.4 Σύνοψη κεφαλαίου 45
Κεφάλαιο 4: Ανάλυση Συστήματος 46
4.1 Αρχιτεκτονική & πρωτόκολλο επικοινωνίας 46
4.2 Το λογισμικό του μικροελεγκτή ESP-32 49
4.3 Το λογισμικό του εξυπηρετητή 53
4.4 Η εφαρμογή για έξυπνες κινητές συσκευές 62
4.4.1 Λειτουργικότητα εφαρμογής 62
4.4.2 Κώδικας εφαρμογής 68
4.5 Η διαδικτυακή εφαρμογή διαχειριστή 70
4.5.1 Λειτουργικότητα εφαρμογής διαχειριστή 70
4.5.2 Κώδικας εφαρμογής διαχειριστή 73
4.6 Η κατασκευή της συσκευής 75
4.6.1 Η βάση της συσκευής 76
4.6.2 Τροφοδοσία του μικροελεγκτή ESP32 77
4.6.4 Σύνδεση με το Ρελέ 79

4.6.5 Σύνδεση διακοπτών 80
4.7 Ασφάλεια συστήματος 80
4.8 Σύνοψη κεφαλαίου 81
Κεφάλαιο 5: Πειραματική Επαλήθευση 82
5.1 Πειράματα 82
5.2 Συμπέρασμα 84
5.3 Σύνοψη κεφαλαίου 84
Κεφάλαιο 6: Επίλογος 85
6.1 Ανακεφαλαίωση 85
6.2 Μετρικά συστήματος 85
6.2.1 Μετρικά υλικού μέρους 86
6.2.2 Μετρικά λογισμικού μέρους 86
6.2.3 Το κόστος του συστήματος 86
6.3 Ανάλυση SWOT 87
6.4 Εκτίμηση κλιμάκωσης συστήματος 88
6.5 Μελλοντικές επεκτάσεις 88
6.6 Προβλήματα που αντιμετωπίστηκαν 89
6.7 Τα στάδια της έρευνας 90
6.8 Συμπεράσματα 90
Παράρτημα – Εγκατάσταση Λογισμικού 92
Βιβλιογραφία 96

Κατάλογος Εικόνων

Εικόνα 1 : Η έξυπνη πρίζα Tapo P110 της TP-LINK....19
Εικόνα 2 : Η έξυπνη πρίζα της Amazon....19
Εικόνα 3 : Η έξυπνη πρίζα της Shelly....20
Εικόνα 4 : Οπτικοποίηση αιτήματος-απάντησης στο πρωτόκολλο HTTP....22
Εικόνα 5 : Οπτικοποίηση λειτουργίας του πρωτοκόλλου MQTT....23
Εικόνα 6 : Μοντέλο της δομής GATT....24
Εικόνα 7 : Διάγραμμα επικοινωνίας πελάτη-εξυπηρετητή με το πρωτόκολλο Modbus RTU....24
Εικόνα 8 : Δομή μηνύματος στο πρωτόκολλο Modbus RTU....25
Εικόνα 9 : Απλό σενάριο Javascript για χρήση στον περιηγητή....26
Εικόνα 10 : Παράδειγμα δημιουργίας HTTP Server σε Node.js....26
Εικόνα 11 : Παράδειγμα κώδικα στο πλαίσιο εφαρμογών React Native....28
Εικόνα 12 : Ο μικροελεγκτής ESP-32....31
Εικόνα 13 : Ο μετρητής κατανάλωσης SDM120M....32
Εικόνα 14 : Ο μετατροπέας TTL σε RS485....33
Εικόνα 15 : Το τροφοδοτικό HI-LINK....33
Εικόνα 16 : Ρελέ 5v....34
Εικόνα 17 : Σχεσιακό διάγραμμα της βάσης δεδομένων....40
Εικόνα 18 : Διάγραμμα επικοινωνίας οντοτήτων με το πρωτόκολλο MQTT....46
Εικόνα 19 : Διάγραμμα επικοινωνίας οντοτήτων με το πρωτόκολλο HTTP....47
Εικόνα 20 : Μηνύματα MQTT....48
Εικόνα 21 : Διάγραμμα ακολουθίας, της επικοινωνίας του χρήστη και της συσκευής με τον διακομιστή....49
Εικόνα 22 : Κώδικας αρχικοποίησης Bluetooth Low Energy....49
Εικόνα 23 : Συνάρτηση αρχικοποίησης WiFi....50
Εικόνα 24 : Κώδικας αρχικοποίησης επικοινωνίας με το μετρητή κατανάλωσης....50
Εικόνα 25 : Κώδικας λήψης της μέτρησης κατανάλωσης....50
Εικόνα 26 : Αποστολή καταγραφής κατανάλωσης στο διακομιστή μέσω του πρωτοκόλλου MQTT....51
Εικόνα 27 : Κώδικας για την λήψη της ημερομηνίας και ώρας από το διακομιστή....51
Εικόνα 28 : Κώδικας για την λήψη και ανάλυση των μηνυμάτων MQTT....51
Εικόνα 29 : Κώδικας αρχικοποίησης εισόδων κουμπιών....52
Εικόνα 30 : Κώδικας για τον έλεγχο του πατήματος των κουμπιών της συσκευής....52
Εικόνα 31 : Αρχικοποίηση των Web Services....53
Εικόνα 32 : Κώδικας για την σύνδεση του χρήστη στο σύστημα....53

Εικόνα 33 : Κώδικας για την αποφυγή επιθέσεων brute force....54
Εικόνα 34 : Μοντέλο χρήστη με την βιβλιοθήκη κώδικα Mongoose.js....59
Εικόνα 35 : Κλήση της συνάρτησης αρχικοποίησης MQTT Server....59
Εικόνα 36 : Δημιουργία MQTT Server με την βιβλιοθήκη κώδικα Aedes.js....59
Εικόνα 37 : Παράδειγμα κώδικα για την σύνδεση στον MQTT Server....60
Εικόνα 38 : Κώδικας ακρόασης γεγονότος (event listener) σύνδεσης συσκευής....60
Εικόνα 39 : Κώδικας ακρόασης γεγονότος (event listener) αποσύνδεσης συσκευής....61
Εικόνα 40 : Αποθήκευση κατάστασης συσκευής σε δομή key-value....61
Εικόνα 41 : Ανακατεύθυνση των αιτημάτων HTTP σε HTTPS....61
Εικόνα 42 : Οθόνη σύνδεσης....63
Εικόνα 43 : Οθόνη εγγραφής....63
Εικόνα 44 : Αρχική οθόνη συνδεδεμένου χρήστη....64
Εικόνα 45 : Οθόνη ομάδων συσκευών....65
Εικόνα 46 : Οθόνη προσθήκης ομάδας συσκευών....65
Εικόνα 47 : Διαγραφή ομάδας συσκευών....65
Εικόνα 48 : Οθόνη προσθήκης συσκευής....66
Εικόνα 49 : Οθόνη στατιστικών συσκευής....66
Εικόνα 50 : Οθόνη επεξεργασίας συσκευής....67
Εικόνα 51 : Οθόνη δημιουργίας γραφημάτων....67
Εικόνα 52 : Οθόνη γραφημάτων....67
Εικόνα 53 : Οθόνη επεξεργασίας στοιχείων χρήστη....68
Εικόνα 54 : Οθόνη κοινής χρήσης συσκευών....68
Εικόνα 55 : Παράδειγμα κώδικα component στο React Native....69
Εικόνα 56 : Κώδικας για την ενεργοποίηση/απενεργοποίηση συσκευής μέσω MQTT....69
Εικόνα 57 : Κώδικας για την αποστολή των στοιχείων σύνδεσης του δίκτυο WiFi στην συσκευή....70
Εικόνα 58 : Οθόνη σύνδεσης στην εφαρμογή διαχειριστή....71
Εικόνα 59 : Οθόνη στατιστικών στην εφαρμογή διαχειριστή....71
Εικόνα 60 : Οθόνη χρηστών στην εφαρμογή διαχειριστή....72
Εικόνα 61 : Οθόνη λεπτομερειών χρήστη στην εφαρμογή διαχειριστή....72
Εικόνα 62 : Οθόνη συμβάντων στην εφαρμογή διαχειριστή....73
Εικόνα 63 : Οθόνη επεξεργασίας στοιχείων λογαριασμού στην εφαρμογή διαχειριστή....73
Εικόνα 64 : Κώδικα για την σύνδεση του χρήστη στην εφαρμογή διαχειριστή....74
Εικόνα 65 : Κώδικας για την αλλαγή της οθόνης μετά από επιτυχημένη σύνδεση....74
Εικόνα 66 : Κώδικας για την σύνδεση του διαχειριστή με την χρήση JWT....75
Εικόνα 67 : Συνδεσμολογία της συσκευής....76
Εικόνα 68 : Υλοποιημένη συσκευή, χωρίς κάλυμμα....76
Εικόνα 69 : Φωτογραφία της συσκευής που υλοποιήθηκε, από εμπρόσθια όψη....77
Εικόνα 70 : Φωτογραφία της συσκευής που υλοποιήθηκε, από πλάγια όψη....77
Εικόνα 71 : Συνδεσμολογία για την τροφοδοσία του ESP-32 με το τροφοδοτικό Hi-Link 78
Εικόνα 72 : Συνδεσμολογία του ESP-32 με το μετρητή κατανάλωσης SDM120M....78
Εικόνα 73 : Διάτρητη πλακέτα στήριξης των ESP-32 και TTL σε RS485....79
Εικόνα 74 : Συνδεσμολογία ESP-32 με το Ρελέ....79
Εικόνα 75 : Σχηματικό σύνδεσης διακοπτών με τον μικροελεγκτή ESP-32....80
Εικόνα 76 : Γράφημα κατανάλωσης ψυγείου....82
Εικόνα 77 : Φωτογραφία ετικέτας χαρακτηριστικών βραστήρα....83
Εικόνα 78 : Πείραμα μέτρησης κατανάλωσης βραστήρα....83
Εικόνα 79 : Πείραμα μέτρησης κατανάλωσης ηλεκτρικής σκούπας....84
Εικόνα 80 : Οπτικοποίηση της τεχνικής αποθήκευσης δεδομένων Bucketing....89

Κατάλογος Πινάκων

Πίνακας 1 : Πίνακας τεχνικών χαρακτηριστικών του μικροεπεξεργαστή ESP-32.....32
Πίνακας 2 : Πίνακας δομής του μοντέλου User 41
Πίνακας 3 : Πίνακας δομής του μοντέλου Device 42
Πίνακας 4 : Πίνακας δομής μοντέλου Device Instance.....43
Πίνακας 5 : Πίνακας δομής του μοντέλου Group.....44
Πίνακας 6 : Πίνακας δομής του μοντέλου Consumption 45
Πίνακας 7 : Πίνακας μετρικών υλικού συστήματος.....86
Πίνακας 8 : Πίνακας μετρικών λογισμικού μέρους.....86
Πίνακας 9 : Πίνακας ανάλυσης κόστους της συσκευής 86

Κεφάλαιο 1: Εισαγωγή

Τα τελευταία χρόνια λόγω της ανάπτυξης της τεχνολογίας οι έξυπνες συσκευές γίνονται ολοένα και περισσότερο κομμάτι της καθημερινότητας μας. Οι συσκευές αυτές, συνθέτουν ένα διαδίκτυο των πραγμάτων (Internet of Things), όπως ονομάζεται, και έχουν την δυνατότητα να συλλέγουν και να μεταφέρουν πληροφορίες, συνήθως αυτόνομα χωρίς ανθρώπινη παρέμβαση [1]. Οι συσκευές του διαδικτύου των πραγμάτων, ενσωματώνουν αισθητήρες και υπολογιστικές συσκευές, ενώ έχουν ένα ευρύ πεδίο εφαρμογών όπως είναι οι οικίες, οι επιχειρήσεις, οι δημόσιοι χώροι, ακόμα και πάνω στο ανθρώπινο σώμα ως φορητές συσκευές.

Στο κεφάλαιο αυτό, αναφέρονται οι λόγοι και οι περιπτώσεις στις οποίες οι έξυπνες συσκευές και τα συστήματα Internet of Things (IoT), μπορούν να φανούν χρήσιμα σε θέματα παρακολούθησης και καταγραφής της ενεργειακής κατανάλωσης. Επίσης, εξετάζονται τα προβλήματα που καλείται να επιλύσει η παρούσα εργασία, ενώ γίνεται σύγκριση με παρόμοιες εργασίες.

1.1 Σκοπός του έργου

Η εργασία αυτή, επικεντρώνεται στο σχεδιασμό και την δημιουργία μιας συσκευής-πρίζας IoT μαζί με το απαραίτητο λογισμικό. Πιο συγκεκριμένα, το σύστημα που υλοποιήθηκε δίνει την δυνατότητα στον χρήστη να ελέγχει τις συσκευές του οπουδήποτε και αν βρίσκεται, μέσα από μια κινητή εφαρμογή, καθώς επίσης να ενημερώνεται μέσω στατιστικών στοιχείων και γραφημάτων σχετικά με την κατανάλωση των συσκευών του.

Βάση των παραπάνω είναι προφανές, ότι το σύστημα αυτό μπορεί να διευκολύνει το χρήστη στην καθημερινότητα του, μέσω της δυνατότητας του απομακρυσμένου ελέγχου. Ενώ, το σημαντικότερο χαρακτηριστικό του συστήματος είναι η δυνατότητα καταγραφής της ενεργειακής κατανάλωσης των συσκευών που συνδέονται με την έξυπνη πρίζα. Στόχος είναι, με τις πληροφορίες που μπορεί να παρέχει το σύστημα, να βοηθήσει τον κάτοχο του να κατανοήσει πώς μπορεί καταστήσει το χώρο του, όπως είναι για παράδειγμα το σπίτι ή η επιχείρηση του, πιο αποδοτικό σε θέματα κατανάλωσης ενέργειας που συνεπάγεται την εξοικονόμηση χρημάτων σε βάθος χρόνου.

1.2 Παρόμοιες εργασίες

Στην υποενότητα αυτή, γίνεται αναφορά σε συστήματα παρόμοια με αυτό που υλοποιήθηκε σε αυτό το έργο και συνοψίζονται τα επιμέρους χαρακτηριστικά τους. Η λίστα των συστημάτων αυτών, συμπληρώθηκε έπειτα από έρευνα στο διαδίκτυο.

1.2.1 TP-LINK Tapo P110

Εικόνα 1 : Η έξυπνη πρίζα Tapo P110 της TP-LINK

Ένα παρόμοιο σύστημα, είναι η έξυπνη πρίζα Tapo P110 από την εταιρεία TP-LINK [2] και εμφανίζεται στην Εικόνα 1. Πρόκειται, για μια μικρή σε μέγεθος και απλή σε σχεδιασμό συσκευή, που υποστηρίζει:

- Απομακρυσμένο έλεγχο
- Μέτρηση κατανάλωσης
- Φωνητικό έλεγχο
- Χρονοπρογραμματισμό

Ο χρήστης, ελέγχει την συσκευή αποκλειστικά μέσω της εφαρμογής που προσφέρει η εταιρεία. Πρέπει να σημειωθεί, πως η TP-LINK αναφέρει απόκλιση στην μέτρηση της κατανάλωσης 3-5%[1].

1.2.2 Amazon Smart Plug

Εικόνα 2 : Η έξυπνη πρίζα της Amazon

Ακόμη ένα παρόμοιο σύστημα είναι η έξυπνη πρίζα της εταιρείας Amazon [3], η οποία φαίνεται στην Εικόνα 2. Η συσκευή είναι κομψή και η εγκατάσταση της γρήγορη. Παρέχει:

- Απομακρυσμένο έλεγχο
- Μέτρηση κατανάλωσης
- Χρονοπρογραμματισμό
- Φωνητικό έλεγχο

Ο Έλεγχος της συσκευής γίνεται αποκλειστικά μέσω του Alexa, του ψηφιακού βοηθού της Amazon.

[1] https://www.tp-link.com/us/support/faq/2410/

1.2.3 Shelly Plug

Εικόνα 3 : Η έξυπνη πρίζα της Shelly

Το τελευταίο παρόμοιο σύστημα που παρουσιάζεται στην Εικόνα 3, είναι η έξυπνη πρίζα της εταιρείας Shelly [4]. Σε σύγκριση με τα προηγούμενα συστήματα που αναφέρθηκαν, αποτελεί την φιλικότερη λύση για τον προγραμματιστή. Αυτό συμβαίνει εξαιτίας του ανοικτού API (Application Programming Interface) που διαθέτει, χαρακτηριστικό που καθιστά εύκολη την διασύνδεση του με τρίτα συστήματα. Υποστηρίζει:

- Απομακρυσμένο έλεγχο
- Μέτρηση κατανάλωσης
- Χρονοπρογραμματισμό

1.3 Συμπεράσματα παρόμοιων συστημάτων

Όπως διαπιστώθηκε, όλα τα παρόμοια συστήματα που αναφέρθηκαν, έχουν χαρακτηριστικά που τα καθιστούν ανταγωνιστικά στην αγορά. Οι δυνατότητες τους συνοψίζονται στην υποστήριξη:

- Απομακρυσμένου ελέγχου, μέσω της κατάλληλης εφαρμογής που παρέχει η κάθε εταιρεία
- Μέτρησης κατανάλωσης ενέργειας, με διατήρηση ιστορικών δεδομένων
- Χρονοπρογραμματισμός ενεργοποίησης και απενεργοποίησης
- Φωνητικές εντολές, μέσω των ψηφιακών βοηθών Google και Alexa
- Ανοικτό API, για την χρήση της πρίζας μέσω άλλων συστημάτων

1.4 Σύνοψη κεφαλαίου

Στο πρώτο κεφάλαιο της διπλωματικής εργασίας, έγινε μια σύντομη εισαγωγή του συστήματος που υλοποιήθηκε. Στην συνέχεια, αναφέρθηκε η χρησιμότητα ενός τέτοιου συστήματος IoT και το εύρος των πιθανών εφαρμογών του. Τέλος, παρουσιάστηκαν συστήματα παρόμοια με το παρόν και συνοψίστηκαν τα χαρακτηριστικά που αυτά διαθέτουν.

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό, παρουσιάζεται το θεωρητικό υπόβαθρο στο οποίο βασίστηκε η ανάπτυξη του συστήματος. Αρχικά, αναφέρονται τα πρωτόκολλα, και στην συνέχεια τα λογισμικά και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του, ενώ στο τέλος τα υλικά μέρη από τα οποία αποτελείται η συσκευή.

2.1 Πρωτόκολλα & τεχνολογίες που χρησιμοποιήθηκαν

Σε αυτή την ενότητα, αναφέρονται τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στην παρούσα εργασία. Πιο συγκεκριμένα, γίνεται μια ιστορική αναδρομή στα πρωτόκολλα HTTP, HTTPS, MQTT, BLE και Modbus, καθώς και μια σύντομη αναφορά στον τρόπο λειτουργίας και χρήσης τους.

2.1.1 HTTP & HTTPS

Το HTTP (HyperText Transfer Protocol), δηλαδή το Πρωτόκολλο Μεταφοράς Υπερκειμένου, είναι το βασικό πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές για την μεταφορά δεδομένων ανάμεσα σε εξυπηρετητή και πελάτη. Η ιδέα του πρωτοκόλλου προτάθηκε από τον Τιμ Μπέρνερς Λι μαζί με την δημιουργία της γλώσσας HTML (HyperText Markup Language) [5].

Η βασική διαδικασία που ακολουθεί το HTTP όπως φαίνεται και στην Εικόνα 4 είναι: (1) σύνδεση στον εξυπηρετητή, (2) αίτημα στον εξυπηρετητή, (3) απάντηση από τον εξυπηρετητή. Οι πιθανές μέθοδοι για αίτημα προς τον εξυπηρετητή είναι οι εξής:

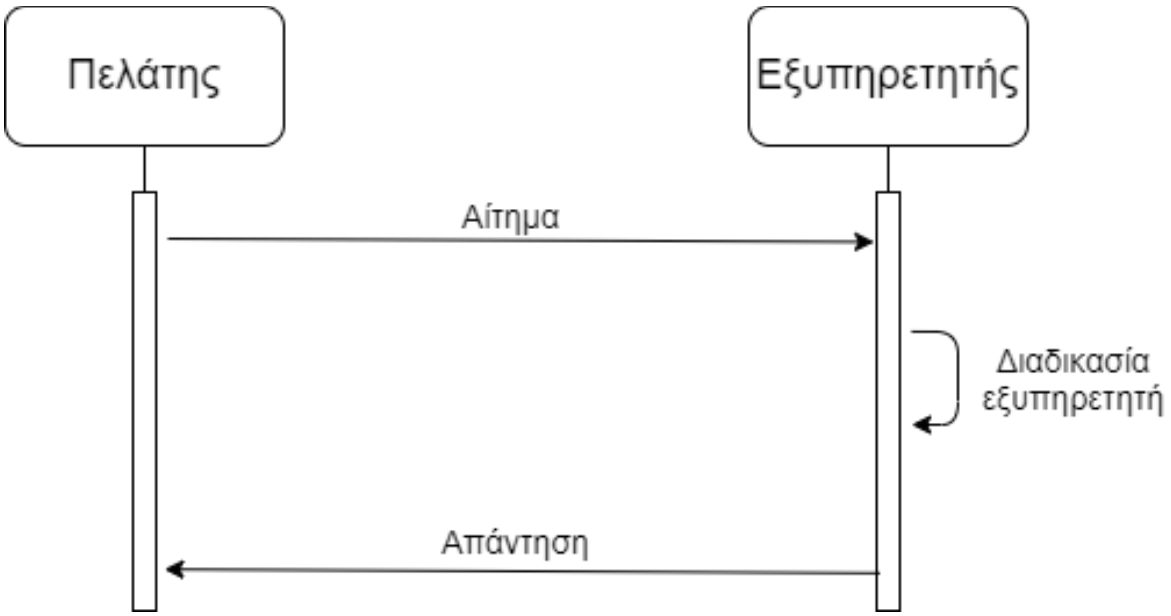
- GET : αίτημα για συγκεκριμένο πόρο. Αυτή η μέθοδος πρέπει να χρησιμοποιείται μόνο για λήψη δεδομένων.
- POST : αίτημα για υποβολή δεδομένων στον διακομιστή.
- HEAD : μοιάζει με την μέθοδο GET αλλά αιτείται μόνο την κεφαλίδα του μηνύματος.
- PUT : η μέθοδος αυτή αντικαθιστά όλο το περιεχόμενο του πόρου στον οποίο στοχεύει με αυτό του αιτήματος.
- DELETE : αίτημα διαγραφής ενός πόρου.
- OPTIONS : αίτημα για το ποιες είναι οι πιθανές μέθοδοι που μπορούν να χρησιμοποιηθούν σε έναν πόρο.

- CONNECT : αίτημα για σύνδεση σε έναν διακομιστή.
- TRACE : αυτή η μέθοδος αιτείται, να επιστρέψει ο διακομιστής την αίτηση. Χρησιμοποιείται για αποσφαλμάτωση.
- PATCH : αίτημα για τροποποίηση ενός πόρου.

Επίσης, κάθε απάντηση από τον εξυπηρετητή συνοδεύεται και από έναν κωδικό κατάστασης ο οποίο δηλώνει αν το αίτημα του πελάτη εξυπηρετήθηκε. Οι διαθέσιμοι κωδικοί κατάστασης είναι οι εξής:

- 100-199 : ενημερωτικές απαντήσεις.
- 200-299 : επιτυχείς απαντήσεις.
- 300-399 : ανακατευθύνσεις (redirects).
- 400-499 : σφάλμα πελάτη.
- 500-599 : σφάλμα διακομιστή.

Το πρωτόκολλο HTTPS (Hypertext Transfer Protocol Secure) χρησιμοποιείται όπως και το HTTP, με την διαφορά ότι η επικοινωνία με τον διακομιστή γίνεται κρυπτογραφημένα. Συγκεκριμένα, το HTTPS δεν διαφέρει από το HTTP, αλλά πρόκειται για τον συνδυασμό του HTTP με το πρωτόκολλο SSL (Secure Sockets Layer), το οποίο χρησιμοποιεί ένα πιστοποιητικό δημοσίου κλειδιού υπογεγραμμένο από μια αρχή πιστοποίησης [6].



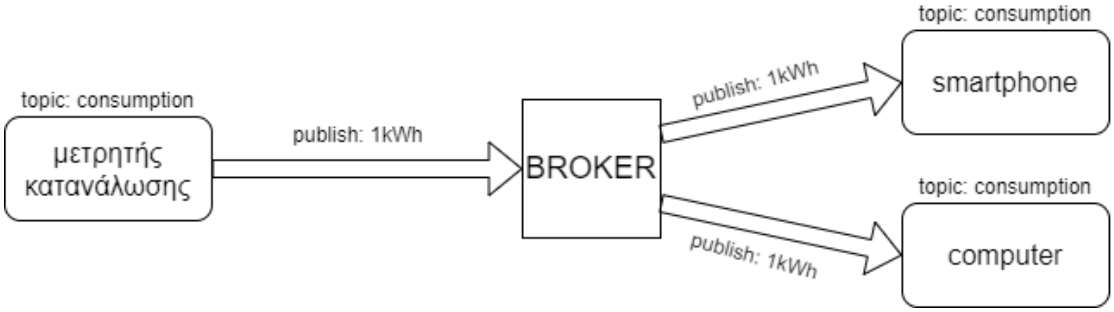
Εικόνα 4 : Οπτικοποίηση αιτήματος-απάντησης στο πρωτόκολλο HTTP

2.1.2 MQTT

Το MQTT (Message Queuing Telemetry Transport), είναι ένα πρωτόκολλο δημοσίευσης-εγγραφής (publish-subscribe) και χρησιμοποιείται για την μεταφορά μηνυμάτων ανάμεσα σε διαφορετικές συσκευές. Η πρώτη έκδοση του δημιουργήθηκε από τους Andy Stanford-Clark και Arlen Nipper το 1991. Σήμερα το MQTT είναι το βασικό πρωτόκολλο επικοινωνίας για συσκευές του διαδικτύου των πραγμάτων (IoT), καθώς μπορεί να χρησιμοποιηθεί σε μικροελεγκτές και σε δίκτυα με πολύ μικρό εύρος ζώνης [7].

Στο MQTT, τρία είναι τα βασικά στοιχεία-οντότητες. Ο διαμεσολαβητής-εξυπηρετητής (broker-server), ο οποίος διαχειρίζεται τα μηνύματα που αποστέλλονται, ο αποστολέας του μηνύματος (publisher) ο οποίος στέλνει το μήνυμα, και ο παραλήπτης (subscriber) ο οποίος το λαμβάνει. Οποιαδήποτε συσκευή είναι συνδεδεμένη στον εξυπηρετητή, μπορεί είτε να στέλνει μηνύματα, είτε να λαμβάνει με την προϋπόθεση ότι είναι εγγεγραμμένη στο κατάλληλο θέμα (topic).

Στην Εικόνα 5 παρουσιάζεται μια απλή περίπτωση επικοινωνίας με το πρωτόκολλο MQTT, κατά την οποία μια συσκευή ενημερώνει μια άλλη σχετικά με την ενέργεια που καταναλώνει.



Εικόνα 5 : Οπτικοποίηση λειτουργίας του πρωτοκόλλου MQTT

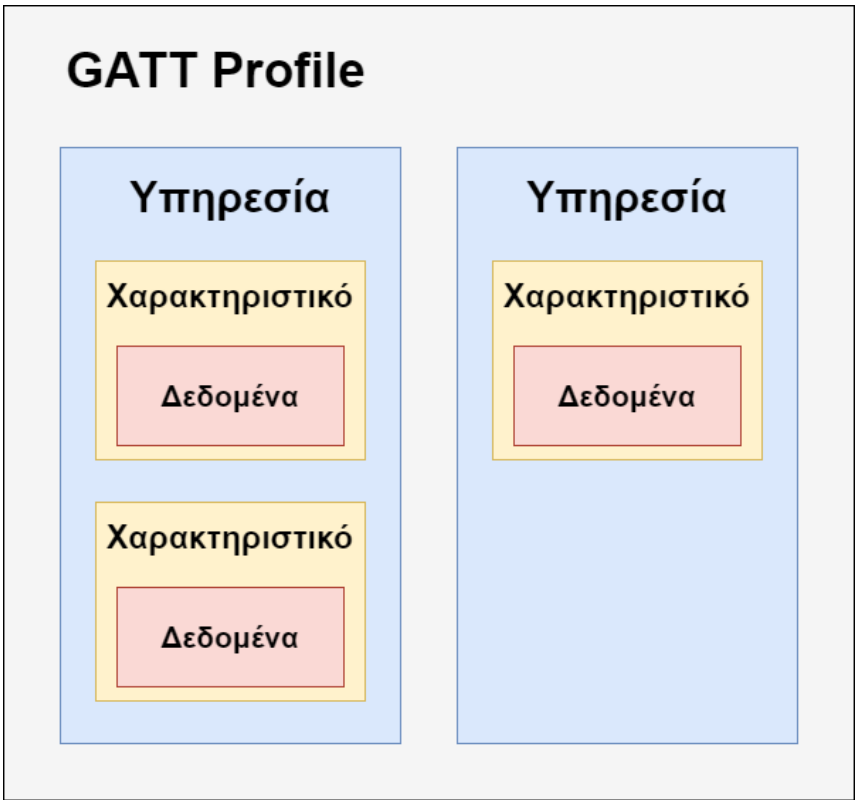
2.1.3 BLE - Bluetooth

Το Bluetooth Low Energy, είναι μια τεχνολογία χαμηλής κατανάλωσης, για την επικοινωνία μεταξύ συσκευών σε μικρές αποστάσεις. Η συσκευή που θα μεταδώσει τα δεδομένα αναλαμβάνει το ρόλο του εξυπηρετητή (server), ενώ η συσκευή που θα τα λάβει έχει το ρόλο του πελάτη (client).

Η ανταλλαγή της πληροφορίας πραγματοποιείται χρησιμοποιώντας μια ιεραρχική δομή δεδομένων, η οποία ονομάζεται προφίλ γενικών γνωρισμάτων (Generic ATTribute profile - GATT) [8]. Το GATT περιλαμβάνει υπηρεσίες που περιέχουν κάποια χαρακτηριστικά, τα οποία εκθέτουν μικρά πακέτα πληροφορίας στον πελάτη, όπως φαίνεται στην Εικόνα 6. Τα πακέτα αυτά μπορούν να είναι, είτε τύπου εγγραφής (write), είτε τύπου ανάγνωσης (read). Πρέπει να σημειωθεί, πως κάθε υπηρεσία και χαρακτηριστικό ορίζεται με ένα καθολικό μοναδικό αναγνωριστικό (UUID).

Μια τυπική επικοινωνία με BLE, ακολουθεί τα εξής βήματα:

- Σάρωση για συσκευή-εξυπηρετητή (server)
- Σύνδεση στην συσκευή-εξυπηρετητή
- Ανακάλυψη των υπηρεσιών της
- Ανακάλυψη των χαρακτηριστικών της κάθε υπηρεσίας
- Ανάγνωση ή εγγραφή πληροφορίας στα χαρακτηριστικά ενδιαφέροντος
- Αποσύνδεση από την συσκευή

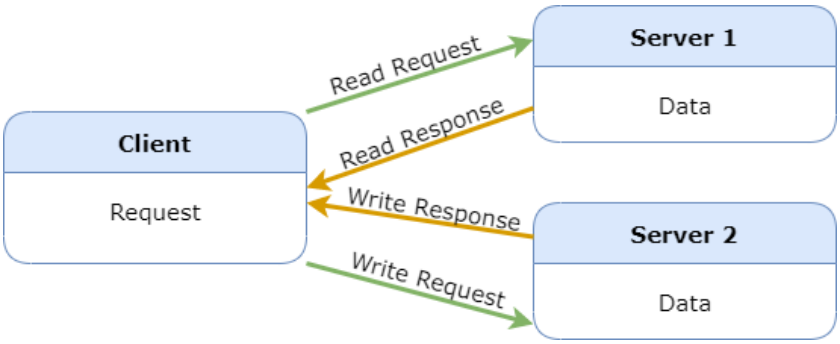


Εικόνα 6 : Μοντέλο της δομής GATT

2.1.4 Modbus RTU

Το πρωτόκολλο Modbus RTU είναι ένα σειριακό πρωτόκολλο, το οποίο υλοποιεί την αρχιτεκτονική πελάτη/εξυπηρετητή και αναπτύχθηκε από την εταιρεία Modicon (σημερινή Schneider Electric) το 1979 [9]. Το Modbus RTU χρησιμοποιείται ευρέως σε βιομηχανικές εφαρμογές, λόγω της απλότητας του στην χρήση και της αξιοπιστίας του.

Για την χρήση του πρωτοκόλλου Modbus RTU απαιτούνται, τουλάχιστον ένας πελάτης (client) και ένας ή περισσότεροι εξυπηρετητές (servers). Στην διεξαγωγή μιας επικοινωνίας Modbus RTU, ο πελάτης εκκινεί ένα αίτημα ανάγνωσης ή εγγραφής προς τον εξυπηρετητή, ο οποίος απαντά με το κατάλληλο μήνυμα. Ένα απλό διάγραμμα επικοινωνίας με Modbus RTU φαίνεται στην Εικόνα 7.



Εικόνα 7 : Διάγραμμα επικοινωνίας πελάτη-εξυπηρετητή με το πρωτόκολλο Modbus RTU

Η αποστολή των μηνυμάτων πραγματοποιείται μέσω 8-bit σειριακής επικοινωνίας, με το κάθε μήνυμα να έχει μέγεθος ως και 256 bytes. Από τα 256 bytes, η μονάδα διεύθυνσης καταλαμβάνει 1 byte, το μήνυμα ως και 253 bytes, ενώ τα 2 τελευταία bytes χρησιμοποιούνται για την ανίχνευση σφαλμάτων με την μέθοδο του κυκλικού ελέγχου πλεονασμού (cyclic redundancy check). Η δομή των μηνυμάτων στο Modbus RTU φαίνεται στην Εικόνα 8.

Μήνυμα Modbus RTU		
Unit Address	***Μήνυμα***	***Cyclic Redundancy Check***
1 Byte	Ως 253 Bytes	2 Bytes

Εικόνα 8 : Δομή μηνύματος στο πρωτόκολλο Modbus RTU

2.2 Λογισμικό μέρος

Η ενότητα αυτή, αναφέρεται στις τεχνολογίες και τα λογισμικά που αξιοποιήθηκαν για την δημιουργία της παρούσας εργασίας. Όλες οι τεχνολογίες και τα λογισμικά που αναφέρονται, υπόκεινται σε άδειες ανοιχτού λογισμικού, δωρεάν χρήσης ή σε άδειες δωρεάν ακαδημαϊκής χρήσης.

2.2.1 Η γλώσσα προγραμματισμού Javascript

Η γλώσσα προγραμματισμού Javascript (JS), δημιουργήθηκε από τον Brendan Eich στην εταιρεία Netscape και κυκλοφόρησε το 1995 [10]. Είναι μια διερμηνευόμενη γλώσσα σεναρίων (scripting) καθοδηγούμενη από συμβάντα (event-driven). Αρχικά, χρησιμοποιήθηκε στους περιηγητές ιστού (web browsers) για την εκτέλεση σεναρίων από την μεριά του πελάτη (client-side), με σκοπό να κάνουν μια σελίδα διαδραστική αλλάζοντας το περιεχόμενο της ασύγχρονα. Ένα παράδειγμα τέτοιας χρήσης είναι στην Εικόνα 9. Η Javascript μαζί με την γλώσσα σήμανσης HTML, η οποία ελέγχει την δομή μια σελίδας, και την γλώσσα CSS, που ελέγχει την εμφάνιση της, αποτελούν την βάση του παγκόσμιου ιστού.

Σήμερα η χρήση της γλώσσας Javascript δεν περιορίζεται μόνο στον περιηγητή. Πιο συγκεκριμένα, χρησιμοποιείται ευρέως για την ανάπτυξη συστημάτων εξυπηρετητών (π.χ. Node.js), εφαρμογών επιτραπέζιου υπολογιστή (π.χ. Electron framework), αλλά και κινητών εφαρμογών (π.χ. React Native, Cordova). Έτσι λοιπόν, εξαιτίας της δυνατότητας που προσφέρει η Javascript για ανάπτυξη λογισμικού για πολλές διαφορετικές πλατφόρμες, είναι ιδιαίτερα δημοφιλής.

```
<script>
    var variable = 'world';
    console.log('hello '+variable);
</script>
```

Εικόνα 9 : Απλό σενάριο Javascript για χρήση στον περιηγητή

2.2.2 Το περιβάλλον εκτέλεσης Node.js

Το Node.js, δημιουργήθηκε από τον Ryan Dahl το 2009. Είναι ένα ανοιχτού κώδικα και ανεξαρτήτου πλατφόρμας (cross platform) περιβάλλον εκτέλεσης (runtime environment) Javascript εκτός του περιηγητή και χρησιμοποιείται για την δημιουργία εργαλείων γραμμής εντολών, καθώς και για την ανάπτυξη διαδικτυακών εφαρμογών [11].

Αντίθετα με τα περισσότερα περιβάλλοντα ανάπτυξης διαδικτυακών εφαρμογών, το περιβάλλον της Node.js δεν βασίζεται στο μοντέλο της πολυνηματικότητας, αλλά σε ένα μοντέλο ασύγχρονων συμβάντων. Το γεγονός αυτό καθιστά την Node.js ιδανική για την ανάπτυξη διαδικτυακών εφαρμογών με δυνατότητα κλιμάκωσης.

Στην Εικόνα 10, φαίνεται ένα σενάριο δημιουργίας ενός απλού HTTP εξυπηρετητή (server) στο περιβάλλον του Node.js.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Εικόνα 10 : Παράδειγμα δημιουργίας HTTP Server σε Node.js

2.2.3 Το πλαίσιο εφαρμογής Express.js

Το Express.js δημιουργήθηκε το 2010 από τον TJ Holowaychuk και είναι ένα διαδικτυακό πλαίσιο εφαρμογής (framework), για το περιβάλλον εκτέλεσης Node.js. Παρέχει μια απλή και ευέλικτη δομή για την ανάπτυξη λογισμικού εξυπηρετητών (servers), ενώ περιλαμβάνει μια εύρωστη συλλογή από χρήσιμα χαρακτηριστικά με σκοπό την γρήγορη δημιουργία διαδικτυακών εφαρμογών [12].

Παρακάτω, γίνεται μια συνοπτική αναφορά στα επιμέρους πακέτα και βιβλιοθήκες λογισμικού που αξιοποιήθηκαν σε συνδυασμό με το Express.js, για την ανάπτυξη του παρών συστήματος στο περιβάλλον του Node.js.

Mongoose.js

Το Mongoose είναι μια βιβλιοθήκη μοντελοποίησης δεδομένων αντικειμένων (Object Data Modeling), για την βάση δεδομένων MongoDB και το Node.js [13]. Χρησιμεύει στην διαχείριση των συσχετίσεων ανάμεσα στα δεδομένα, όπως και στην επικύρωση των δεδομένων πριν αυτά εισαχθούν στην βάση MongoDB.

MQTT – Aedes

Το Aedes είναι μια βιβλιοθήκη λογισμικού για το Node.js, η οποία υλοποιεί το πρωτόκολλο ανταλλαγής μηνυμάτων MQTT για επικοινωνία μεταξύ διαφορετικών συσκευών [14]. Το Aedes, έχει το ρόλο του εξυπηρετητή-διαμεσολαβητή στην επικοινωνία αυτή, προωθώντας τα μηνύματα που λαμβάνει στα κατάλληλα κανάλια.

JSON Web Tokens

Τα JSON Web Tokens (JWT), είναι ένα πρότυπο το οποίο καθορίζει ένα συμπαγή και ασφαλή τρόπο μετάδοσης πληροφορίας ανάμεσα σε δύο οντότητες, με την μορφή ενός αντικειμένου JSON (JavaScript Object Notation) [15]. Στο σύστημα που περιγράφεται, τα JWT χρησιμοποιούνται για την αυθεντικοποίηση των χρηστών. Συγκεκριμένα, για την δημιουργία του JWT, χρησιμοποιείται το αναγνωριστικό του χρήστη στην βάση, καθώς και η διεύθυνση IP του. Τυπικά, ένα JWT έχει την μορφή : xxxxx.yyyyy.zzzzz, με το πρώτο μέρος να είναι η κεφαλίδα, που περιέχει πληροφορίες για τον τύπο του και τον αλγόριθμο υπογραφής (signing algorithm) που χρησιμοποιήθηκε. Το δεύτερο μέρος περιέχει τα δεδομένα, ενώ το τρίτο την υπογραφή του JWT, η οποία περιλαμβάνει και ένα μυστικό κλειδί. Τέλος, κάθε τμήμα του JWT κωδικοποιείται στη μορφή base64.

Bcrypt

Ο Bcrypt είναι ένας αλγόριθμος κατακερματισμού (hashing) και σχεδιάστηκε από τους Niels Provos και David Mazières το 1999, ενώ βασίστηκε στην κρυπτογράφηση Blowfish [16]. Χρησιμοποιώντας την κατάλληλη βιβλιοθήκη λογισμικού, στο παρόν σύστημα ο αλγόριθμος αυτός αξιοποιείται για τον κατακερματισμό των κωδικών πρόσβασης των χρηστών, ώστε να αποθηκευτούν στην βάση δεδομένων με ασφάλεια.

2.2.4 Το πλαίσιο κινητών εφαρμογών React Native

Το React Native είναι ένα πλαίσιο εφαρμογών (framework) για κινητές συσκευές, που δημιουργήθηκε από την εταιρεία Meta Platforms (πρώην Facebook) το 2015 [17]. Βασίζεται στο πλαίσιο εφαρμογών React, το οποίο χρησιμεύει στην δημιουργία διεπαφών χρήστη με την γλώσσα προγραμματισμού Javascript. Το React Native χρησιμοποιείται για την δημιουργία εφαρμογών για τις πλατφόρμες Android, IOS, Android TV, tv OS, MacOS, Windows αλλά και για UWP (Universal Windows Platform).

Συνδυάζοντας το React μαζί με τρίτες βιβλιοθήκες λογισμικού, το React Native επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές που προορίζονται για πολλές διαφορετικές πλατφόρμες, με μια κοινή βάση κώδικα γραμμένη στην γλώσσα προγραμματισμού Javascript.
Στην Εικόνα 11 που ακολουθεί, φαίνεται ένα απλό παράδειγμα κώδικα γραμμένο στο πλαίσιο εφαρμογών React Native με το γραφικό του αποτέλεσμα.

Εικόνα 11 : Παράδειγμα κώδικα στο πλαίσιο εφαρμογών React Native

Συνδυαστικά με το React Native χρησιμοποιήθηκαν και κάποιες βιβλιοθήκες λογισμικού. Παρακάτω, αναφέρονται συνοπτικά το όνομα και η λειτουργία των σημαντικότερων από αυτές.

React Navigation
Το React Navigation είναι μια βιβλιοθήκη λογισμικού, η οποία χρησιμεύει στην εναλλαγή διαφορετικών οθονών μέσα στην εφαρμογή [18]. Υλοποιεί μια δομή τύπου στοίβας (stack navigator), στην οποίο ο χρήστης μπορεί να προσθέτει οθόνες (push) και να τις αφαίρει (pop), επιτρέποντας του να κατευθύνεται προς τα πίσω.

React Native Async Storage
Το async storage είναι μια βιβλιοθήκη για React Native, η οποία υλοποιεί ένα σύστημα αποθήκευσης του μοντέλου κλειδίου-τιμής (key-value) [19]. Χρησιμοποιείται για την αποθήκευση δεδομένων του χρήστη και συγκεκριμένα αποθηκεύει τοπικά το αλφαριθμητικό JWT, για την αυτόματη σύνδεση του στο σύστημα.

React Native MQTT
Το React Native MQTT είναι μια βιβλιοθήκη λογισμικού, που αξιοποιείται για την σύνδεση της κινητής εφαρμογής με τον MQTT Server και την αποστολή και λήψη μηνυμάτων [20] προς και από την συσκευή.

React Native BLE plx
Το React Native BLE plx είναι μια βιβλιοθήκη λογισμικού, η οποία δίνει την δυνατότητα σε μια εφαρμογή να ανακαλύψει συσκευές Bluetooth στον χώρο, καθώς και να διαβάσει ή να γράψει δεδομένα από και προς τις συσκευές αυτές [21].

2.2.5 Η γλώσσα προγραμματισμού C/C++

Η C είναι μια διαδικαστική γλώσσα προγραμματισμού γενικής χρήσης, η οποία αναπτύχθηκε από τον Ντένις Ρίτσι και εμφανίστηκε πρώτη φορά το 1972 [22]. Η γλώσσα C++ είναι ένα υπερσύνολο της C με την βασική τους διαφορά να είναι η αντικειμενοστρέφεια. Αναπτύχθηκε το 1979 από τον Μπιάρνε Στρούστρουπ, με σκοπό την επέκταση της C [23].

Στο έργο αυτό, οι C και C++ χρησιμοποιήθηκαν για τον προγραμματισμό του μικροεπεξεργαστή ESP-32. Για την χρήση των C και C++, ήταν αναγκαία η εγκατάσταση της συλλογής ανάπτυξης λογισμικού (SDK) της εταιρείας ESPRESSIF[2]. Επιπλέον, χρησιμοποιήθηκαν οι εξής βιβλιοθήκες ανοιχτού κώδικα:

- WiFi - Είναι μια προεγκατεστημένη βιβλιοθήκη, η οποία δίνει την δυνατότητα στον μικροεπεξεργαστή να συνδεθεί σε ένα δίκτυο WiFi, να σαρώσει το χώρο για τα διαθέσιμα δίκτυα WiFi, αλλά και να δημιουργήσει ένα φορητό σημείο πρόσβασης για συνδεθούν άλλες συσκευές.
- BLE – Είναι και αυτή μια προεγκατεστημένη βιβλιοθήκη λογισμικού, η οποία μπορεί να συμπεριληφθεί ώστε να δώσει την δυνατότητα στον μικροεπεξεργαστή ESP-32 να σαρώσει το χώρο για συσκευές BLE (Bluetooth Low Energy), να δημιουργήσει ένα διακομιστή (Server) BLE ή να δημιουργήσει ένα πελάτη (Client) BLE.
- Preferences – Αποτελεί προεγκατεστημένη βιβλιοθήκη κώδικα, η οποία επιτρέπει την μόνιμη αποθήκευση δεδομένων με την μορφή κλειδιού-τιμής (key-value).
- HTTPClient – Είναι μια προεγκατεστημένη βιβλιοθήκη λογισμικού, που επιτρέπει την αποστολή αιτημάτων HTTP, είτε με την μέθοδο GET, είτε με την μέθοδο POST.
- Arduino_JSON – Σε συνδυασμό με την βιβλιοθήκη HTTPClient, η βιβλιοθήκη λογισμικού Arduino_JSON χρησιμοποιείται για την ανάλυση των μηνυμάτων τύπου JSON που αποστέλλει ο διακομιστής.
- SDM – Η βιβλιοθήκη SDM αποτελεί ένα πακέτο λογισμικού, που επιτρέπει την αποστολή και λήψη μηνυμάτων με το πρωτόκολλο Modbus. Η βιβλιοθήκη, αφού ρυθμιστούν κατάλληλα οι διάφορες παράμετροι της, απλοποιεί την επικοινωνία με συσκευές που υποστηρίζουν Modbus [24].

[2] https://www.espressif.com/en/support/download/sdks-demos

2.2.6 Η βάση δεδομένων MongoDB

Για την αποθήκευση των δεδομένων του συστήματος, χρησιμοποιήθηκε η βάση δεδομένων MongoDB. Η MongoDB κρίθηκε κατάλληλη για λόγους οι οποίοι σχετίζονται με την επεκτασιμότητα και την απόδοση.

Πιο συγκεκριμένα, η βάση δεδομένων MongoDB εκτός από κάθετη επέκταση (vertical scaling), δηλαδή προσθήκη μνήμης και επεξεργαστικής ισχύος, υποστηρίζει και οριζόντια επέκταση (horizontal scaling) με δύο τρόπους. Η πρώτη μέθοδος είναι ο τεμαχισμός (sharding), η οποία επιτρέπει την αξιοποίηση πολλαπλών κόμβων. Έτσι αυξάνεται ο όγκος δεδομένων που μπορεί να διεκπεραιωθεί, ιδιαίτερα σε θέματα εγγραφής (write). Η δεύτερη μέθοδος είναι τα σετ αντιγράφων (replica sets). Με την επέκταση αυτού του τύπου, τα δεδομένα αντιγράφονται σε πολλαπλούς κόμβους, έχοντας ως αποτέλεσμα αυξημένη ασφάλεια και δυνατότητα γρηγορότερης εκτέλεσης ενεργειών ανάγνωσης (read) [25].

2.3 Εργαλεία

Σε αυτή την ενότητα, παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα της κινητής εφαρμογής, της εφαρμογής διαχειριστή, του εξυπηρετητή και της συσκευής. Όλα τα εργαλεία που αναφέρονται, παρέχονται δωρεάν.

2.3.1 Visual Studio Code

Το visual studio code [26], είναι ένας ανοιχτού κώδικα επεξεργαστής κειμένου που αναπτύχθηκε από την εταιρεία Microsoft για τα λειτουργικά συστήματα Windows, Linux και macOS.

Το βασικό του χαρακτηριστικό, είναι η ποικιλία από πρόσθετα που είναι διαθέσιμα προς εγκατάσταση. Έτσι, μπορεί κάποιος να προσθέσει επιπλέον χαρακτηριστικά και λειτουργίες όπως για παράδειγμα υποστήριξη πολλών διαφορετικών γλωσσών προγραμματισμού, εργαλεία αποσφαλμάτωσης και αυτόματη στοίχιση κώδικα. Το γεγονός αυτό, καθιστά το visual studio code εύκολα προσαρμόσιμο στις ανάγκες κάθε προγραμματιστή.

2.3.2 Arduino IDE

Το Arduino IDE [27](Integrated Development Environment) είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα για τους μικροελεγκτές Arduino. Διαθέτει τα ελάχιστα απαραίτητα χαρακτηριστικά για την συγγραφή κώδικα, ενώ έχει ενσωματωμένο μεταγλωττιστή (compiler) και την δυνατότητα να ανεβάσει τον εκτελέσιμο κώδικα στον μικροελεγκτή.

Εκτός της υποστήριξης για την πλατφόρμα του Arduino, με εγκατάσταση των κατάλληλων εργαλείων το Arduino IDE μπορεί να υποστηρίξει και μικροελεγκτές διαφορετικών κατασκευαστών.

2.3.3 Android Studio

Το Android Studio [28] αποτελεί ένα εργαλείο βασισμένο στο Intellij IDEA, της εταιρείας Jetbrains, και αποτελεί ένα προγραμματιστικό περιβάλλον για την ανάπτυξη εφαρμογών android. Το Android Studio είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0.

2.4 Υλικό μέρος

Παρακάτω, γίνεται αναφορά στο υλικό που χρησιμοποιήθηκε για την δημιουργία αυτής της εργασίας. Συγκεκριμένα, οι υποενότητες που ακολουθούν αφορούν τον μικροελεγκτή που χρησιμοποιήθηκε, τον αισθητήρα ρεύματος, το ρελέ και το τροφοδοτικό της συσκευής.

2.4.1 Ο Μικροελεγκτής ESP-32

Εικόνα 12 : Ο μικροελεγκτής ESP-32

Το ESP-32 [29] είναι ένα μικροελεγκτής, ο οποίος φαίνεται στην Εικόνα 12, και ενδείκνυται για εφαρμογές διαδικτύου των πραγμάτων (IoT). Ο λόγος είναι ότι υποστηρίζει Wi-Fi και Bluetooth που το καθιστά ικανό για ασύρματη σύνδεση στο διαδίκτυο και για άμεση επικοινωνία με άλλες συσκευές, ενώ παράλληλα έχει χαμηλή κατανάλωση ενέργειας και χαμηλό κόστος. Τα αναλυτικά χαρακτηριστικά του μικροεπεξεργαστή ESP-32 φαίνονται στον παρακάτω πίνακα (Πίνακας 1). Περισσότερες λεπτομέρειες σχετικά με τα χαρακτηριστικά του ESP-32 βρίσκονται στο φύλλο δεδομένων του (data sheet).[3]

[3] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Μικροεπεξεργαστής	Xtensa LX6 32-bit
Ψηφιακές εισόδοι-εξόδοι	25
Αναλογικές εξόδοι	2
Αναλογικές εισόδοι	6
UARTs	3
Μνήμη Flash	4 MB
Μνήμη SRAM	520 KB
Τάση λειτουργίας	3.3 V
Πρωτόκολλο WiFi	IEEE 802.11 b/g/n
Πρωτόκολλο Bluetooth	Bluetooth 4.2

Πίνακας 1 : Πίνακας τεχνικών χαρακτηριστικών του μικροεπεξεργαστή ESP-32

2.4.2 Ο μετρητής κατανάλωσης SDM120M

Εικόνα 13 : Ο μετρητής κατανάλωσης SDM120M

Ο SDM120M [30] είναι ένας οικονομικός μετρητής κατανάλωσης από την εταιρεία Eastron και εμφανίζεται στην Εικόνα 13. Συγκεκριμένα, ο μετρητής παρέχει μετρήσεις για την τάση και την ένταση του ρεύματος, την συχνότητα, την ισχύ και την κατανάλωση. Επιπλέον, υποστηρίζει δύο διεπαφές. Μια έξοδο παλμού για την κοινοποίηση της κατανάλωσης σε τρίτες συσκευές, με τον κάθε παλμό να ισοδυναμεί με 0,001 kWh, και μία διεπαφή RS485 για σειριακή επικοινωνία. Για την διεπαφή RS485 χρησιμοποιείται το πρωτόκολλο Modbus. Μερικές ενδεικτικές διευθύνσεις Modbus φαίνονται παρακάτω και μπορούν να βρεθούν στο έγγραφο ορισμού πρωτοκόλλου του SDM120M[4].

- 0x0000 - Τάση ρεύματος
- 0x0006 - Ένταση ρεύματος
- 0x000c - Ενεργή ισχύς
- 0x0046 - Συχνότητα ρεύματος

Τέλος, στο φύλλο δεδομένων του μετρητή SDM120M, αναφέρεται ακρίβεια μέτρησης με απόκλιση 1%[5]. Το γεγονός αυτό, καθιστά τον μετρητή εξαιρετικά ακριβή και κατάλληλο για την συγκεκριμένη εργασία.

[4] https://www.eastroneurope.com/images/uploads/products/protocol/SDM120-MODBUS_Protocol.pdf
[5] https://www.eastroneurope.com/images/uploads/products/manuals/SDM120_Series_Manual_.pdf

2.4.3 Ο μετατροπέας TTL σε RS485

Εικόνα 14 : Ο μετατροπέας TTL σε RS485

Για να επιτευχθεί η επικοινωνία του μικροελεγκτή ESP32 με το μετρητή κατανάλωσης SDM120, χρησιμοποιείται ένας μετατροπέας από TTL (Transistor-Transistor Logic) σε RS485, οποίος φαίνεται στην Εικόνα 14. Ενσωματώνει το ολοκληρωμένο κύκλωμα MAX485, για να μετατρέπει την σειριακή διεπαφή του μικροελεγκτή στο πρότυπο RS485. Ο μετατροπέας μπορεί να χρησιμοποιηθεί για την επικοινωνία μέχρι και 32 συσκευών σε μέγιστη απόσταση 1,2 χιλιόμετρα, με ταχύτητες μετάδοσης έως 10Mbit/s. Τα λεπτομερή χαρακτηριστικά του MAX485 συναντώνται στο φύλλο δεδομένων του.[6]

2.4.4 Το τροφοδοτικό HI-LINK

Εικόνα 15 : Το τροφοδοτικό HI-LINK

Η συσκευή που χρησιμοποιείται για την τροφοδότηση του μικροελεγκτή είναι το τροφοδοτικό HI-LINK, το οποίο φαίνεται στην Εικόνα 15 και μετατρέπει τα 220v εναλλασσόμενου ρεύματος σε 5v συνεχούς. Ο μικρός του όγκος και το χαμηλό του κόστος το καθιστά ιδανικό για συσκευές IoT, άρα και για την συγκεκριμένη εργασία. Λεπτομέρειες για τα χαρακτηριστικά του τροφοδοτικού HI-LINK βρίσκονται στο φύλλο δεδομένων του.[7]

[6] https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf
[7] https://datasheet.lcsc.com/szlcsc/1909111105_HI-LINK-HLK-PM24_C399250.pdf

2.4.5 Το ρελέ

Εικόνα 16 : Ρελέ 5v

Για την ενεργοποίηση και απενεργοποίηση της παροχής ρεύματος της συσκευής, χρησιμοποιήθηκε ένα ρελέ με τάση πηνίου τα 5 volt και ικανό να ελέγξει ως 250 volt και 10 Ampere. Το ρελέ που χρησιμοποιήθηκε φαίνεται στην Εικόνα 16. Τα λοιπά χαρακτηριστικά του ρελέ, μπορούν να βρεθούν στο φύλλο δεδομένων του.[8]

2.5 Σύνοψη κεφαλαίου

Σε αυτό το κεφάλαιο, αναφέρθηκαν οι τεχνολογίες, το λογισμικό και το υλικό που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας πτυχιακής εργασίας. Επιπλέον, έγινε μια σύντομη ανάλυση στις βασικές αρχές λειτουργίας τους και τα χαρακτηριστικά, που τα καθιστούν κατάλληλα για την χρήση τους στο παρόν έργο. Το επόμενο κεφάλαιο, θα αναφερθεί στις λειτουργικές απαιτήσεις του συστήματος που υλοποιήθηκε και τον τρόπο που αλληλοεπιδρούν οι χρήστες με αυτό.

[8] https://handsontec.com/dataspecs/relay/1Ch-relay.pdf

Κεφάλαιο 3: Απαιτήσεις Συστήματος

Στο προηγούμενο κεφάλαιο, παρουσιάστηκαν οι τεχνολογίες, τα λογισμικά, καθώς και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος. Στο παρόν κεφάλαιο, μελετώνται οι απαιτήσεις και οι προδιαγραφές που τηρήθηκαν για την εξασφάλιση της λειτουργικότητας του συστήματος, αλλά και της ασφάλειας των χρηστών του.

3.1 Απαιτήσεις συστήματος

Για να επιτευχθούν οι στόχοι και η λειτουργικότητα ενός συστήματος, είναι χρήσιμο να οριστούν οι απαιτήσεις του. Να αναφερθούν δηλαδή, με σαφή τρόπο οι προδιαγραφές που πρέπει να τηρηθούν, ώστε να είναι εύχρηστο και λειτουργικό. Οι παρακάτω απαιτήσεις διαμορφώθηκαν στο στάδιο της έρευνας.

Σχεδιασμός εφαρμογής (UI/UX)

Ο σχεδιασμός και η υλοποίηση της κινητής εφαρμογής πρέπει να γίνουν με τρόπο, που να την καθιστούν εύχρηστη και ευχάριστη. Έχοντας κατά νου ότι πολλοί χρήστες δεν έχουν μεγάλη εμπειρία στη χρήση κινητών συσκευών, κάθε στοιχείο της εφαρμογής πρέπει να είναι απλό σε σχεδιασμό και να "περιγράφει" τη λειτουργία του, είτε με κείμενο, είτε με ένα κατάλληλο εικονίδιο, ενώ η πρόσβαση στις βασικές λειτουργίες της εφαρμογής θα γίνεται από την αρχική οθόνη. Επίσης, τα κουμπιά θα πρέπει να έχουν τέτοιο μέγεθος, ώστε ο χρήστης να τα επιλέγει εύκολα με τη πρώτη προσπάθεια.

Σχεδιασμός συσκευής

Η συσκευή θα πρέπει να χρειάζεται ελάχιστες ρυθμίσεις από το χρήστη, ώστε η εκκίνηση της λειτουργίας της να είναι άμεση και χωρίς κόπο (plug and play). Επιπλέον, ο όγκος της πρέπει είναι όσο το δυνατόν μικρότερος για να είναι βολική και εύκολη στην μεταφορά.

Ασφάλεια συστήματος

Ιδιαίτερα σημαντικό είναι κατά την υλοποίηση του έργου, να δοθεί επαρκής σημασία στην ασφάλεια. Η επικοινωνία των συσκευών και της εφαρμογής με το διακομιστή, πρέπει να γίνεται κρυπτογραφημένα και σύμφωνα με όλα τα σύγχρονα πρότυπα ασφαλείας, ενώ ισχυρή κρυπτογράφηση θα χρησιμοποιηθεί και για την αποθήκευση των κωδικών των χρηστών.

Ασφάλεια χρήστη

Απαραίτητη προϋπόθεση, είναι να εξασφαλιστεί η ασφάλεια και η σωματική ακεραιότητα των χρηστών του συστήματος. Συγκεκριμένα, κάθε συσκευή που συνδέεται στο ηλεκτρικό δίκτυο θα πρέπει να είναι γειωμένη και μονωμένη, ενώ επίσης είναι σημαντικό να υπάρχει και κύκλωμα προστασίας από βραχυκυκλώματα, για την αποφυγή πυρκαγιάς.

Επεκτασιμότητα & συντήρηση

Το σύστημα πρέπει να υποστηρίζει πολλαπλές συσκευές και πολλαπλούς χρήστες. Παράλληλα, ο κώδικας πρέπει να είναι επεκτάσιμος και κατανοητός, ώστε να είναι εύκολη η συντήρηση του και οι μελλοντικές αναβαθμίσεις.

3.2 Περιπτώσεις χρήσης

Στις υποενότητες που ακολουθούν, παρουσιάζονται οι περιπτώσεις χρήσης αφού κάποιος εγκαταστήσει την εφαρμογή στο έξυπνο κινητό του τηλέφωνο(smartphone), αλλά και οι δυνατότητες που έχει ο διαχειριστής από το διαδικτυακό περιβάλλον του συστήματος μέσω του περιηγητή(browser). Στο σύστημα διακρίνονται 4 είδη χρηστών: (1) μη συνδεδεμένοι χρήστες, (2) συνδεδεμένοι χρήστες, (3) διαχειριστές και (4) συσκευές.

3.2.1 Μη συνδεδεμένος χρήστης

Οι παρακάτω περιπτώσεις αφορούν την κινητή εφαρμογή:

Περίπτωση χρήσης 1: Εγγραφή

Σε αυτή την περίπτωση, ο χρήστης δεν είναι εγγεγραμμένος στο σύστημα και για να συνεχίσει θα πρέπει να δημιουργήσει λογαριασμό εισάγοντας email και κωδικό.

Περίπτωση χρήσης 2: Επιβεβαίωση της διεύθυνσης ηλεκτρονικού ταχυδρομείου

Ο χρήστης αφού έχει κάνει εγγραφή στο σύστημα, θα πρέπει να μεταβεί στη διεύθυνση ηλεκτρονικού ταχυδρομείου (email) του, ώστε να επιβεβαιώσει την εγγραφή του.

Περίπτωση χρήσης 3: Σύνδεση

Σε αυτή την περίπτωση, ο χρήστης είναι ήδη εγγεγραμμένος στο σύστημα και για να συνεχίσει θα πρέπει συνδεθεί με την ηλεκτρονική του διεύθυνση(email) και τον κωδικό του, ώστε να αποκτήσει πρόσβαση στις συσκευές του.

3.2.2 Συνδεδεμένος χρήστης

Οι παρακάτω περιπτώσεις αφορούν την κινητή εφαρμογή:

Περίπτωση χρήσης 1: Αποθήκευση διαπιστευτηρίων

Αμέσως μετά την σύνδεση του χρήστη στην εφαρμογή, αποθηκεύονται τοπικά στην συσκευή τα διαπιστευτήρια του, ώστε να μην χρειαστεί να συνδεθεί ξανά εισάγοντας τα στοιχεία του για ένα μικρό χρονικό διάστημα.

Περίπτωση χρήσης 2: Προσθήκη ομάδας συσκευών
Ο συνδεδεμένος χρήστης μπορεί να μεταβεί στην οθόνη ομάδων συσκευών. Εκεί, έχει την δυνατότητα να προσθέσει μια ομάδα συσκευών εισάγοντας ένα όνομα ή και μια περιγραφή για την ομάδα και να την αποθηκεύσει.

Περίπτωση χρήσης 3: Επεξεργασία ομάδας συσκευών
Από την οθόνη ομάδων συσκευών, ο χρήστης μπορεί να επιλέξει μια ομάδα και να επεξεργαστεί το όνομα και την περιγραφή της.

Περίπτωση χρήσης 4: Αφαίρεση ομάδας συσκευών
Από την οθόνη ομάδων συσκευών, ο χρήστης έχει την δυνατότητα να αφαιρέσει/διαγράψει μια ομάδα συσκευών.

Περίπτωση χρήσης 5: Προσθήκη συσκευής
Ο χρήστης, από την αρχική οθόνη, μπορεί να μεταβεί στην οθόνη προσθήκης συσκευής. Εκεί πρέπει να εισάγει ένα όνομα για τη συσκευή, να επιλέξει εικονίδιο και χρώμα και προαιρετικά μια ομάδα συσκευών. Επιπλέον, πρέπει να συνδεθεί με την συσκευή μέσω Bluetooth, ώστε να της γνωστοποιήσει τα στοιχεία πρόσβασης του ασύρματου δικτύου(WIFI) και να λάβει το μοναδικό αναγνωριστικό(id) της συσκευής.

Περίπτωση χρήσης 6: Επεξεργασία συσκευής
Επιλέγοντας μια συσκευή, ο χρήστης μεταβαίνει στην οθόνη της συσκευής όπου μπορεί να συνεχίσει στην οθόνη επεξεργασίας των στοιχείων της συσκευής. Εκεί, έχει τη δυνατότητα να αλλάξει το όνομα, το εικονίδιο και το χρώμα της.

Περίπτωση χρήσης 7: Αφαίρεση συσκευής
Στην οθόνη επεξεργασίας της συσκευής, υπάρχει κατάλληλο κουμπί για την αφαίρεση/διαγραφή της.

Περίπτωση χρήσης 8: Έλεγχος συσκευής
Στον χρήστη παρέχεται η δυνατότητα να ενεργοποιήσει ή να απενεργοποιήσει οποιαδήποτε από τις συσκευές του μέσω διαδικτύου.

Περίπτωση χρήσης 9: Ενημέρωση κατανάλωσης συσκευής
Ο χρήστης πατώντας πάνω στη συσκευή, μπορεί να ενημερωθεί για την ενέργεια που έχει αυτή καταναλώσει μέσα σε συγκεκριμένα χρονικά διαστήματα. Επιπλέον, υπάρχει μια ένδειξη των Watt που καταναλώνει η συσκευή σε πραγματικό χρόνο.

Περίπτωση χρήσης 10: Προβολή γραφημάτων
Στην οθόνη γραφημάτων, δίνεται η δυνατότητα δημιουργίας, προβολής και λήψης γραφημάτων, ώστε να συγκρίνει την κατανάλωση των διάφορων συσκευών του.

Περίπτωση χρήσης 11: Επεξεργασία στοιχείων χρήστη
Ο χρήστης έχει πρόσβαση στην οθόνη λογαριασμού, όπου μπορεί να δει τα στοιχεία του, αλλά και να αλλάξει το email του ή τον κωδικό του.

Περίπτωση χρήσης 12: Αποσύνδεση χρήστη
Από την οθόνη λογαριασμού, ο χρήστης μπορεί να αποσυνδεθεί από την εφαρμογή.

3.2.3 Διαχειριστής

Στις περιπτώσεις του διαχειριστή περιλαμβάνονται όλες οι περιπτώσεις χρήσης που αφορούν το συνδεδεμένο χρήστη. Οι επιπλέον δυνατότητες που παρέχονται στο διαχειριστή του συστήματος, είναι διαθέσιμες από μια διαδικτυακή εφαρμογή προσβάσιμη από το φυλλομετρητή.

Περίπτωση χρήσης 1: Σύνδεση στον ιστότοπο
Αφού ο διαχειριστής επισκεφτεί τη διεύθυνση της διαδικτυακής εφαρμογής από τον περιηγητή (browser), για να συνεχίσει θα πρέπει να εισάγει τη διεύθυνση ηλεκτρονικού ταχυδρομείου (email) του και τον κωδικό του και να συνδεθεί.

Περίπτωση χρήσης 2: Ενημέρωση κατάστασης συστήματος
Όταν ο διαχειριστής συνδεθεί, η πρώτη οθόνη που βλέπει είναι η οθόνη ελέγχου (dashboard), στην οποία θα μπορεί να ενημερωθεί για τον αριθμό των εγγεγραμμένων χρηστών, το συνολικό αριθμό των συσκευών, τον αριθμό των ενεργών συσκευών και το χρόνο λειτουργίας του συστήματος (uptime).

Περίπτωση χρήσης 3: Προβολή χρηστών
Από τον κατάλογο συνδέσμων (menu) της σελίδας ο διαχειριστής μπορεί να μεταβεί στη σελίδα των χρηστών, στην οποία θα ενημερώνεται για όλους τους χρήστες του συστήματος και το ποιοι από αυτούς έχουν επιβεβαιώσει την εγγραφή τους μέσω ηλεκτρονικού ταχυδρομείου (email).

Περίπτωση χρήσης 4: Λεπτομέρειες χρήστη
Επιλέγοντας κάποιον χρήστη από την οθόνη των χρηστών, ο διαχειριστής έχει τη δυνατότητα να δει διάφορα στοιχεία για το χρήστη όπως ο αριθμός των συσκευών του και πότε συνδέθηκε τελευταία φορά, αλλά και να απενεργοποιήσει το λογαριασμό του.

Περίπτωση χρήσης 5: Ρυθμίσεις λογαριασμού
Από το menu της σελίδα, ο διαχειριστής μπορεί να μεταβεί στη σελίδα λογαριασμού, όπου θα έχει τη δυνατότητα να αλλάξει τη διεύθυνση ηλεκτρονικού ταχυδρομείου (email) του και τον κωδικό του.

Περίπτωση χρήσης 6: Αποσύνδεση
Από το menu της σελίδας, ο διαχειριστής πατώντας στον κατάλληλο σύνδεσμο αποσυνδέεται από το λογαριασμό του.

3.2.4 Συσκευή

Οι παρακάτω περιπτώσεις χρήσης αφορούν την έξυπνη πρίζα η οποία μετρά την κατανάλωση ενέργειας και παρέχει απομακρυσμένο έλεγχο ενεργοποίησης / απενεργοποίησης.

Περίπτωση χρήσης 1: Πρώτη εκκίνηση λειτουργίας
Όταν οι συσκευές κάνουν εκκίνηση της λειτουργίας τους για πρώτη φορά, ενεργοποιούν το Bluetooth και αναμένουν σύζευξη με την κινητή εφαρμογή.

Περίπτωση χρήσης 2: Σύνδεση στο ασύρματο τοπικό δίκτυο
Αφού οι συσκευές λάβουν από την κινητή εφαρμογή το όνομα δικτύου(SSID) και τον κωδικό μέσω του προτύπου Bluetooth, τα χρησιμοποιούν για να συνδεθούν στο ασύρματο τοπικό δίκτυο(WIFI) και να αποκτήσουν πρόσβαση στο διαδίκτυο. Αν δεν καταφέρουν να συνδεθούν μετά από μερικές προσπάθειες, θα γίνεται επανεκκίνηση.

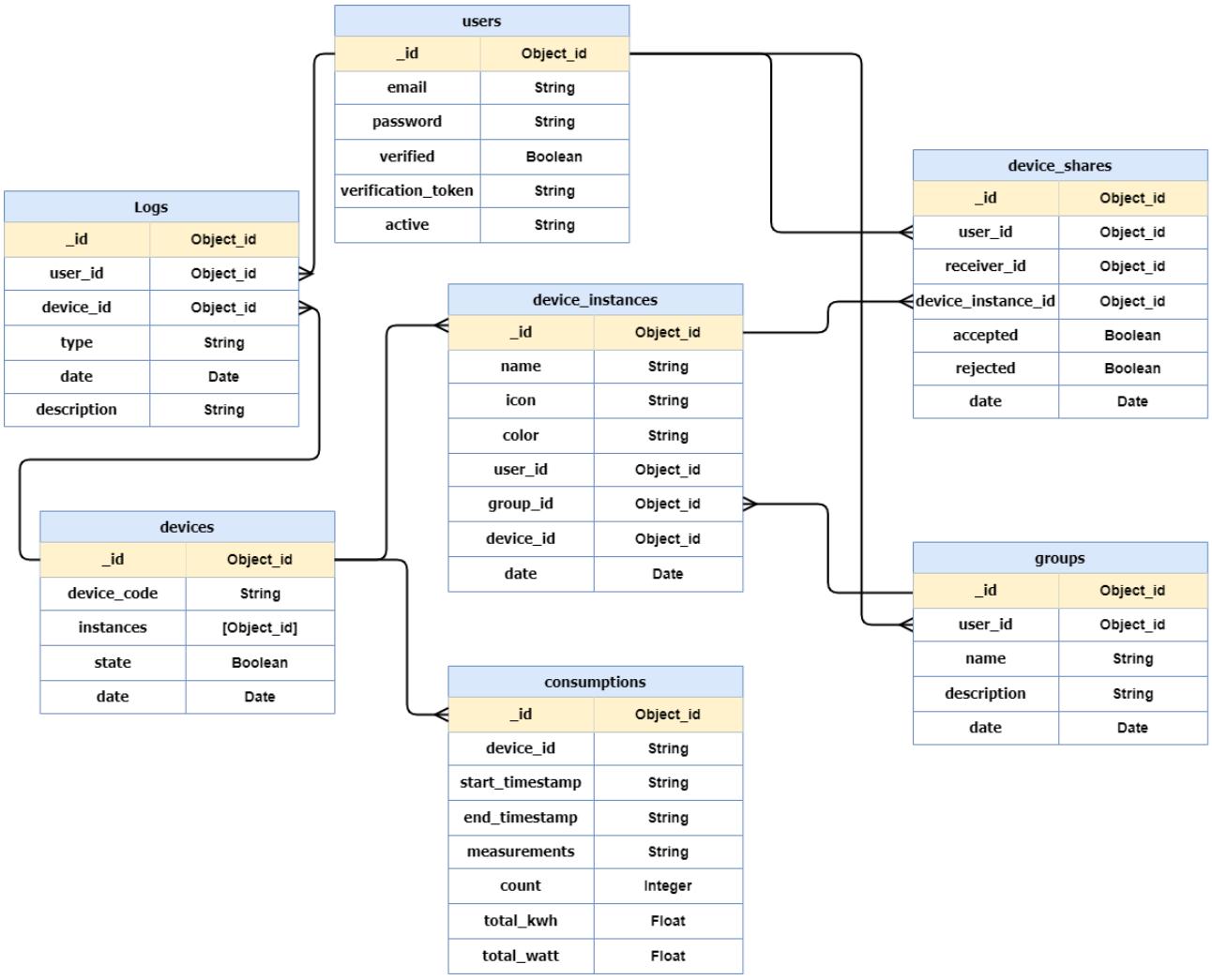
Περίπτωση χρήσης 3: Αποστολή κατανάλωσης
Περιοδικά, οι συσκευές θα ενημερώνουν το διακομιστή για την κατανάλωση ενέργειας που καταγράφουν.

Περίπτωση χρήσης 4: Ενεργοποίηση/Απενεργοποίηση
Η συσκευή επικοινωνεί με το διακομιστή σε πραγματικό χρόνο για την ενεργοποίηση ή απενεργοποίηση της από το χρήστη.

3.3 Βάση δεδομένων

Η βάση δεδομένων MongoDB που χρησιμοποιήθηκε, αποτελεί μια noSQL βάση, το οποίο σημαίνει πως κάθε καταγραφή αντιμετωπίζεται ουσιαστικά σαν ένα αρχείο (document) και ανήκει σε μια συλλογή αρχείων (collection). Το κάθε document δεν έχει προκαθορισμένα πεδία, το οποίο καθιστά τις μελλοντικές ενημερώσεις και αλλαγές στο σύστημα εύκολες, χωρίς διακοπή λειτουργίας(downtime). Αν έπρεπε να συγκριθεί η MongoDB με μια SQL βάση δεδομένων, όπως είναι η MariaDB και η MySQL, το document θα ήταν το αντίστοιχο μιας εγγραφής (record) και το collection θα ήταν το αντίστοιχο της δομής πίνακα (table).

3.3.1 Σχεσιακό διάγραμμα της βάσης δεδομένων



Εικόνα 17 : Σχεσιακό διάγραμμα της βάσης δεδομένων

Στην Εικόνα 17, παρουσιάζεται το σχεσιακό διάγραμμα της βάσης δεδομένων που δημιουργήθηκε, με τις συλλογές της, τα πεδία της και τους τύπους των πεδίων αυτών.

3.3.2 Συλλογές της βάσης δεδομένων

Στην υποενότητα αυτή, αναλύονται οι συλλογές από τις οποίες αποτελείται η βάση δεδομένων και επεξηγούνται τα διάφορα πεδία τους. Πιο συγκεκριμένα, τα πεδία χαρακτηρίζονται με βάση τον τύπο τους, το αν είναι απαραίτητα στην δημιουργία μιας εγγραφής, την μοναδικότητα ή και κάποια επιπλέον χαρακτηρίστηκα που ίσως έχουν.

Παρακάτω, αναλύονται οι συλλογές της βάσης δεδομένων με την βοήθεια σχετικών πινάκων και επεξηγήσεων.

Η συλλογή users

Σε αυτή την συλλογή, αποθηκεύονται τα δεδομένα των χρηστών. Ο Πίνακας 2 απεικονίζει τη δομή του αρχείου καταγραφής (document) ενός χρήστη.

Field	Type	Required	Unique	Extra
_id	objectID	true	true	auto-generated
role	string	true	false	default: user
email	string	true	true	
password	string	true	false	
verification_token	string	false	true	auto-generated
verified	boolean	false	false	default: false
active	boolean	true	false	default: true

Πίνακας 2 : Πίνακας δομής του μοντέλου User

- Το πεδίο **_id** είναι τύπου objectId, το οποίο σημαίνει πως περιέχει: μια τιμή χρονικής σήμανσης (timestamp - η στιγμή της δημιουργίας του), ένα τυχαίο αλφαριθμητικό (string) και μία αυξανόμενη τιμή (auto-increment). Το πεδίο αυτό δημιουργείται αυτόματα για κάθε καταγραφή (document) και είναι μοναδικό.
- Το πεδίο **role** είναι τύπου string. Είναι απαραίτητο και οι πιθανές τιμές του είναι **user** και **admin.** Καθορίζει τα δικαιώματα του χρήστη στο σύστημα.
- Το πεδίο **email** είναι τύπου string. Επίσης είναι απαραίτητο και μοναδικό για κάθε χρήστη.
- Το πεδίο **password** είναι τύπου string. Είναι απαραίτητο για την εγγραφή του χρήστη και δεν είναι μοναδικό. Πριν αποθηκευτεί ο κωδικός (password) κατακερματίζεται (hash) από τον αλγόριθμο bcrypt χρησιμοποιώντας ένα 128-bit salt. Ο bcrypt βασίζεται στο κρυπτογράφημα blowfish.
- Το πεδίο **verification_token** είναι τύπου string και μοναδικό για κάθε χρήστη. Αποστέλλεται ως παράμετρος του url στην ηλεκτρονική διεύθυνση (email) του χρήστη, ώστε να επιβεβαιώσει την εγγραφή του.
- Το πεδίο **verified** είναι τύπου boolean. Η προκαθορισμένη τιμή είναι false και αλλάζει σε true, όταν ο χρήστης επιβεβαιώσει το email του.
- Το πεδίο **active** είναι τύπου boolean και η προκαθορισμένη τιμή του είναι true. Προσδιορίζει το αν κάποιος λογαριασμός είναι ενεργός.

Η συλλογή devices

Σε αυτή την συλλογή, αποθηκεύονται τα δεδομένα των συσκευών. Ο Πίνακας 3 απεικονίζει τη δομή του αρχείου καταγραφής (document) μιας συσκευής.

Field	Type	Required	Unique	Extra
_id	objectID	true	true	auto-generated
device_code	string	true	true	
state	boolean	false	false	default: false
instances	objectID	false	false	array, ref: Device Instance
date	date	false	false	default: creation datetime

Πίνακας 3 : Πίνακας δομής του μοντέλου Device

- Το πεδίο **_id** είναι τύπου objectId, το οποίο σημαίνει πως περιέχει: μια τιμή χρονικής σήμανσης (timestamp - η στιγμή της δημιουργίας του), ένα τυχαίο αλφαριθμητικό (string) και μία αυξανόμενη τιμή (auto-increment). Το πεδίο αυτό δημιουργείται αυτόματα για κάθε καταγραφή (document) και είναι μοναδικό.
- Το πεδίο **device_code** είναι τύπου string. Είναι απαραίτητο και μοναδικό για κάθε συσκευή και αποτελεί ένα τυχαίο αναγνωριστικό το οποίο είναι αποθηκευμένο στην συσκευή.
- Το πεδίο **state** είναι τύπου boolean. Δεν είναι ούτε απαραίτητο ούτε μοναδικό πεδίο. Αφορά το αν η συσκευή είναι ενεργοποιημένη ή απενεργοποιημένη.
- Το πεδίο **instances** είναι ένα πεδίο πίνακας και τύπου objectId. Δεν είναι μοναδικό ή απαραίτητο πεδίο και αποτελεί ξένο κλειδί στον πίνακα device instances.
- Το πεδίο **date** είναι τύπου date. Δεν είναι απαραίτητο πεδίο και δεν είναι μοναδικό. Αποτελεί την στιγμή της δημιουργίας της συσκευής η οποία είναι και η προκαθορισμένη τιμή του.

Η συλλογή device_instances

Στη συλλογή αυτή, αποθηκεύονται τα δεδομένα των παραλλαγών κάθε συσκευής όπως αυτές ρυθμίζονται από κάθε διαφορετικό χρήστη. Ο χρήστης επιλέγει διαφορετικά χαρακτηριστικά σχετικά με το όνομα, την περιγραφή, το χρώμα, την ομάδα συσκευών και το εικονίδιο της συσκευής του. Ο Πίνακας 4 απεικονίζει τη δομή του αρχείου καταγραφής (document) μιας παραλλαγής συσκευής.

Field	Type	Required	Unique	Extra
_id	objectID	true	true	auto-generated
device	objectID	true	false	ref: devices
name	string	true	false	
description	string	false	false	
group	objectID	false	false	ref: groups
icon	string	true	false	
color	string	true	false	
user_id	objectID	true	false	
date	date	false	false	default: creation datetime

Πίνακας 4 : Πίνακας δομής μοντέλου Device Instance

- Το πεδίο **_id** είναι τύπου objectId, το οποίο σημαίνει πως περιέχει: μια τιμή χρονικής σήμανσης (timestamp - η στιγμή της δημιουργίας του), ένα τυχαίο αλφαριθμητικό (string) και μία αυξανόμενη τιμή (auto-increment). Το πεδίο αυτό δημιουργείται αυτόματα για κάθε καταγραφή (document) και είναι μοναδικό.
- Το πεδίο **device** είναι τύπου objectId και αναφέρεται στην μοναδική συσκευή από την συλλογή devices με την οποία σχετίζεται η κάθε καταγραφή. Είναι απαραίτητο αλλά όχι μοναδικό πεδίο.
- Το πεδίο **name** είναι τύπου string. Είναι απαραίτητο αλλά όχι μοναδικό πεδίο και πρόκειται για το όνομα που έχει δώσει ο χρήστης στην συσκευή του.
- Το πεδίο **group** είναι τύπου objectId. Δεν είναι απαραίτητο ούτε μοναδικό και αποτελεί ξένο κλειδί του πίνακα groups.
- Το πεδίο **description** είναι τύπου string. Δεν είναι απαραίτητο ούτε μοναδικό πεδίο και πρόκειται για την περιγραφή που έχει δώσει ο χρήστης στην συσκευή του.
- Το πεδίο **date** είναι τύπου date. Δεν είναι απαραίτητο πεδίο και δεν είναι μοναδικό. Αποτελεί την στιγμή της δημιουργίας της καταγραφής η οποία είναι και η προκαθορισμένη τιμή του.

Η συλλογή groups

Σε αυτή την συλλογή, βρίσκονται τα δεδομένα των ομάδων συσκευών. Ο Πίνακας 5 παρουσιάζει τη δομή του αρχείου καταγραφής (document) μιας ομάδας.

Field	Type	Required	Unique	Extra
_id	objectID	true	true	auto-generated
name	string	true	false	
description	string	false	false	
user_id	string	true	false	

Πίνακας 5 : Πίνακας δομής του μοντέλου Group

- Το πεδίο **_id** είναι τύπου objectId, το οποίο σημαίνει πως περιέχει: μια τιμή χρονικής σήμανσης (timestamp - η στιγμή της δημιουργίας του), ένα τυχαίο αλφαριθμητικό (string) και μία αυξανόμενη τιμή (auto-increment). Το πεδίο αυτό δημιουργείται αυτόματα για κάθε καταγραφή (document) και είναι μοναδικό.
- Το πεδίο **name** είναι τύπου string. Δεν είναι απαραίτητο πεδίο για την εγγραφή της συσκευής αλλά δεν είναι μοναδικό. Είναι ουσιαστικά ένα όνομα το οποίο δίνει ο χρήστης στην συσκευή για να την αναγνωρίζει εύκολα.
- Το πεδίο **description** είναι τύπου string. Δεν είναι απαραίτητο για μια συσκευή και προφανώς δεν είναι μοναδικό. Χρησιμοποιείται για να δώσει ο χρήστης μια περιγραφή στην συγκεκριμένη συσκευή.
- Το πεδίο **user_id** είναι τύπου string και αναφέρεται στο πεδίο _id της συλλογής users.

Η συλλογή consumptions

Στην συλλογή αυτή, γίνεται η αποθήκευση των καταγραφών που σχετίζονται με την ενεργειακή κατανάλωση κάθε συσκευής. Κάθε καταγραφή αντιπροσωπεύει 1 ώρα λειτουργίας της συσκευής και περιέχει υπό-καταγραφές του ενός λεπτού. Ο Πίνακας 6 απεικονίζει τη δομή των αρχείων καταγραφής μιας κατανάλωσης.

Field	Type	Required	Unique	Extra
_id	objectID	true	true	auto-generated
device	objectID	true	false	
start_timestamp	date	true	false	
end_timestamp	date	true	false	
measurements	object	false	false	array
count	number	true	false	default: 0
total_kwh	number	true	false	default: 0

Πίνακας 6 : Πίνακας δομής του μοντέλου Consumption

- Το πεδίο **_id** είναι τύπου objectId, το οποίο σημαίνει πως περιέχει: μια τιμή χρονικής σήμανσης (timestamp - η στιγμή της δημιουργίας του), ένα τυχαίο αλφαριθμητικό (string) και μία αυξανόμενη τιμή (auto-increment). Το πεδίο αυτό δημιουργείται αυτόματα για κάθε καταγραφή (document) και είναι μοναδικό.
- Το πεδίο **device** είναι τύπου objectId και αποτελεί ξένο κλειδί της συλλογής devices.
- Το πεδίο **start_timestamp** είναι τύπου date και πρόκειται για την ώρα της ημέρας (xx:00:00) στην οποία γίνονται οι καταγραφές της κατανάλωσης.
- Το πεδίο **end_timestamp** είναι τύπου date και πρόκειται για το τέλος της ώρας (xx:59:59) στην οποία γίνονται οι καταγραφές της κατανάλωσης.
- Το πεδίο **measurements** είναι ένας πίνακας ο οποίος περιέχει ένα αντικείμενο καταγραφής, το οποίο περιέχει την τιμή της καταγραφής όπως και την στιγμή δημιουργίας της.
- Το πεδίο **count** είναι αριθμός που αντιπροσωπεύει τον πλήθος των υπό-καταγραφών που περιέχεται στο πεδίο measurements.
- Το πεδίο **sum** είναι αριθμός που αντιπροσωπεύει τον άθροισμα των τιμών των υπό-καταγραφών που περιέχεται στο πεδίο measurements.
- Το πεδίο **total_kwh** είναι αριθμός και αντιπροσωπεύει την συνολική κατανάλωση των υπο-καταγραφών που περιέχεται στο πεδίο measurements.

3.4 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό, έγινε μια εκτενής ανάλυση των λειτουργικών απαιτήσεων και των περιπτώσεων χρήσης του συστήματος. Επίσης, παρουσιάστηκε ο τρόπος και η μορφή που τα δεδομένα των χρηστών και των συσκευών, αποθηκεύονται στην βάση δεδομένων, σε ότι αφορά την οργάνωση τους στις διάφορες συλλογές και στα επιμέρους χαρακτηριστικά τους. Στο επόμενο κεφάλαιο, παρουσιάζονται το λογισμικό, η σύνδεση του υλικού μέρους, καθώς και η μεταξύ τους επικοινωνία.

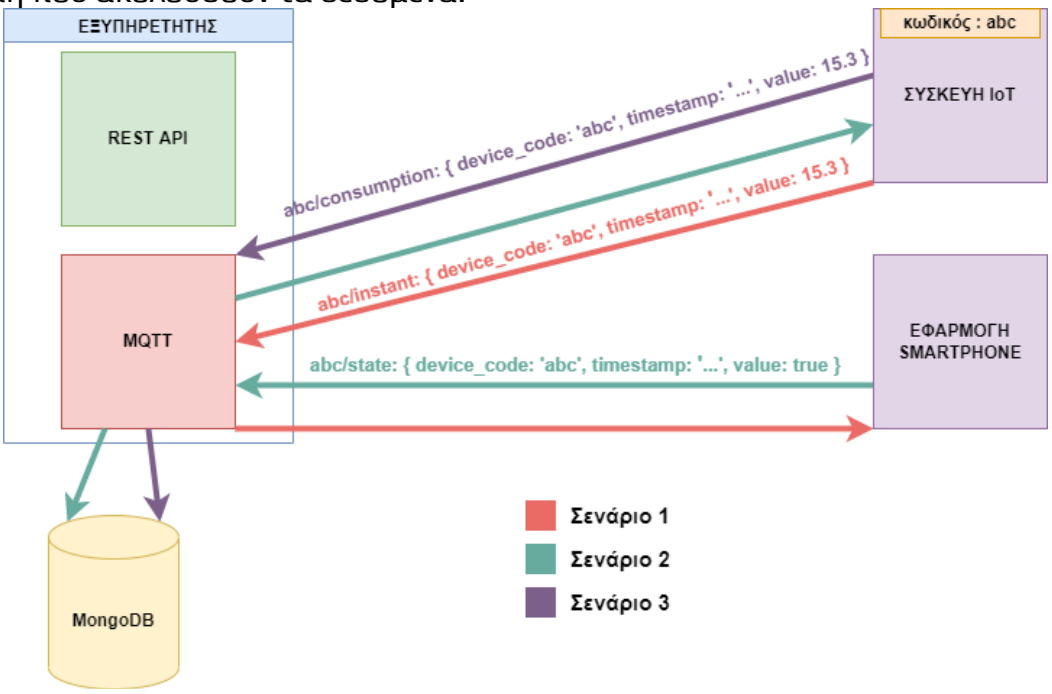
Κεφάλαιο 4: Ανάλυση Συστήματος

Η ενότητα αυτή, αναφέρει τα βασικότερα σημεία του κώδικα που συγγράφτηκε για την υλοποίηση αυτής της εργασίας. Συγκεκριμένα, παρουσιάζονται τμήματα κώδικα της συσκευής, του εξυπηρετητή, της κινητής εφαρμογής και της διαδικτυακής εφαρμογής διαχείρισης. Οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν είναι οι C, C++ και η Javascript.

4.1 Αρχιτεκτονική & πρωτόκολλο επικοινωνίας

Σκοπός της παρούσας ενότητας είναι η ανάλυση και κατανόηση της αρχιτεκτονικής του συστήματος που υλοποιήθηκε. Αυτό επιτυγχάνεται με την χρήση διαγραμμάτων και παραδειγμάτων επικοινωνίας.

Το διάγραμμα στην Εικόνα 18, επικεντρώνεται στην χρήση του πρωτοκόλλου MQTT. Συγκεκριμένα, αναλύονται τρείς περιπτώσεις επικοινωνίας και επεξηγείται η διαδρομή που ακολουθούν τα δεδομένα.



Εικόνα 18 : Διάγραμμα επικοινωνίας οντοτήτων με το πρωτόκολλο MQTT

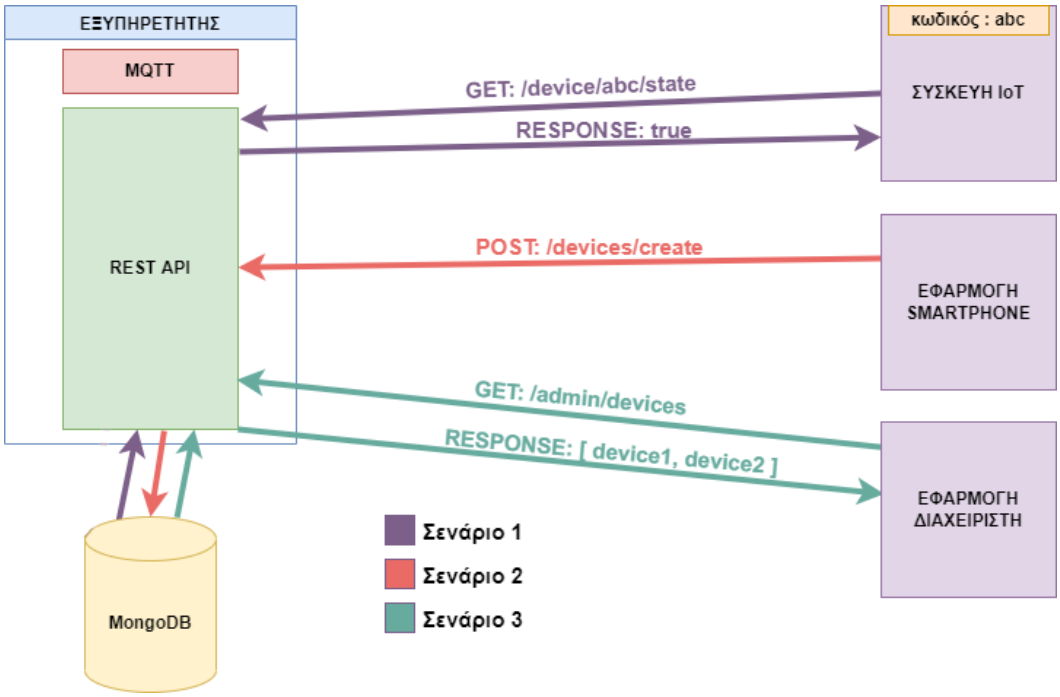
Στο σενάριο 1 του διαγράμματος στην Εικόνα 18, εμφανίζεται η επικοινωνία της συσκευής IoT με την κινητή εφαρμογή για την εμφάνιση της κατανάλωσης σε πραγματικό χρόνο. Η θεωρητική συσκευή με κωδικό 'abc' (ο κωδικός είναι απλοποιημένος για λόγους ευκολίας στο παράδειγμα) δημοσιεύει στο κανάλι 'abc/consumption' ένα μήνυμα που αποτελείται από τον κωδικό της συσκευής, την στιγμή της μέτρησης και την μέτρηση της κατανάλωσης. Στην συνέχεια ο εξυπηρετητής προωθεί το μήνυμα στην κινητή εφαρμογή, η οποία με την σειρά της εμφανίζει την κατανάλωση στην οθόνη του χρήστη.

Στο σενάριο 2 του διαγράμματος, η επικοινωνία ακολουθεί την αντίθετη διαδρομή, με σκοπό την ενεργοποίηση της συσκευής. Ο χρήστης, μέσω της κινητής εφαρμογής, δημοσιεύει στο κανάλι 'abc/state'. Το μήνυμα, το οποίο προωθείται μέσω του εξυπηρετητή στην συσκευή, περιέχει τον κωδικό της συσκευής, την στιγμή της εντολής ενεργοποίησης και την λογική τιμή 'true'. Στην περίπτωση που ο χρήστης αιτούταν την απενεργοποίηση της συσκευής, η λογική τιμή θα ήταν 'false'. Κατά την διάρκεια αυτής της επικοινωνίας, η νέα κατάσταση της συσκευής αποθηκεύεται στην βάση δεδομένων.

Στην τελευταία περίπτωση του διαγράμματος, στο σενάριο 3, παρουσιάζεται η αποστολή της κατανάλωσης της συσκευής και η αποθήκευση της από τον εξυπηρετητή. Τα μηνύματα αυτά δημιουργούνται ανά ένα λεπτό και η τιμή που αποστέλλεται από την συσκευή, είναι ένας μέσος όρος από δειγματοληπτικές μετρήσεις. Κατά την αποθήκευση στην βάση δεδομένων, υπολογίζονται και οι κιλοβατώρες που καταναλώθηκαν. Ο υπολογισμός των κιλοβατώρων για ένα λεπτό λειτουργίας γίνεται ως εξής:

$$y\ kWh = \frac{x\ Watt * \left(\frac{1}{60}\right)}{1000}$$

Στο διάγραμμα στην Εικόνα 19, παρουσιάζονται μερικά σενάρια επικοινωνίας μέσω του REST API του εξυπηρετητή με το πρωτόκολλο HTTP. Τα βέλη δείχνουν την διαδρομή που ακολουθούν τα δεδομένα σε κάθε αίτημα-σενάριο.



Εικόνα 19 : Διάγραμμα επικοινωνίας οντοτήτων με το πρωτόκολλο HTTP

Στο σενάριο 1 εμφανίζεται η διαδρομή ενός αιτήματος HTTP με την μέθοδο GET. Συγκεκριμένα, η συσκευή αιτείται αμέσως μετά την σύνδεση της στο διαδίκτυο, την κατάσταση που έχει ορίσει ο χρήστης, δηλαδή αν είναι ενεργοποιημένη ή απενεργοποιημένη. Ο εξυπηρετητής λαμβάνει το αίτημα και στην συνέχεια ανακτά την κατάσταση της συσκευής από την βάση για να την αποστείλει ως απάντηση στο αίτημα.

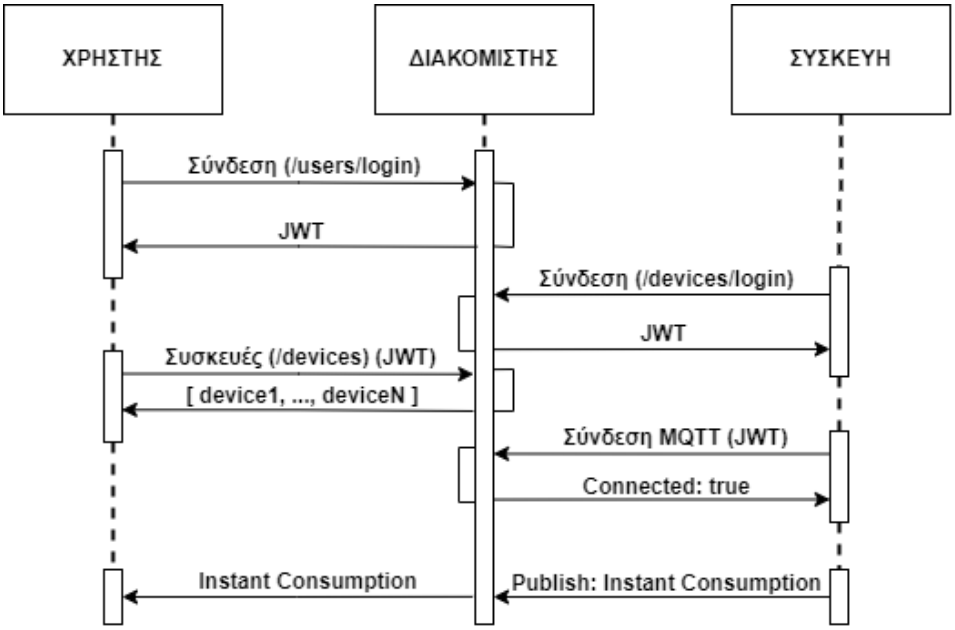
Στο σενάριο 2 του διαγράμματος στην Εικόνα 19, φαίνεται ένα αίτημα δημιουργίας μιας νέας συσκευής με την μέθοδο HTTP POST. Ο χρήστης μέσω της κινητής εφαρμογής στέλνει τα δεδομένα προς αποθήκευση στον εξυπηρετητή, ο οποίος δημιουργεί μια νέα καταγραφή στην βάση δεδομένων.

Τέλος, στο σενάριο 3, παρουσιάζεται το HTTP GET αίτημα του διαχειριστή για την εμφάνιση όλων των συσκευών του συστήματος προς τον εξυπηρετητή, που με την σειρά του φέρνει τα δεδομένα από την βάση και τα επιστρέφει ως απάντηση.

Όπως έγινε φανερό και στα διαγράμματα στην Εικόνα 18, τα μηνύματα που ανταλλάσσονται με το πρωτόκολλο MQTT ακολουθούν συγκεκριμένη μορφή. Η μορφή αυτή περιλαμβάνει πάντα τον κωδικό της συσκευής, ο οποίος είναι ένα μοναδικό αλφαριθμητικό 64 χαρακτήρων, και την χρονική στιγμή που δημιουργείται. Το τρίτο στοιχείο του μηνύματος είναι αυτό που αλλάζει ανάλογα την περίπτωση, και μπορεί να έχει τιμή τύπου Float (δεκαδικός αριθμός) ή τύπου Boolean (λογική τιμή true ή false). Η τιμή τύπου Float, αποστέλλεται στα κανάλια instant και consumption και αφορά μια μέτρηση κατανάλωσης, ενώ η τιμή τύπου Boolean αποστέλλεται στο κανάλι state και σηματοδοτεί την ενεργοποίηση ή απενεργοποίηση της συσκευής. Στην Εικόνα 20 παρουσιάζεται η μορφή των δύο αυτών πιθανών μηνυμάτων.

Εικόνα 20 : Μηνύματα MQTT

Είναι επίσης χρήσιμο, να παρουσιαστεί γραφικά η διαδικασία για την έναρξη της επικοινωνίας ενός χρήστη με τον διακομιστή και η έναρξη της επικοινωνίας μιας συσκευής με τον διακομιστή. Στην Εικόνα 21 παρουσιάζεται το διαγράμματα ακολουθίας για μια τυπική επικοινωνία ενός χρήστη και μιας συσκευής με τον διακομιστή. Όπως φαίνεται, η πρώτη ενέργεια του χρήστη και της συσκευής είναι να συνδεθούν, ώστε να λάβουν το JWT (JSON Web Token). Στην συνέχεια, ο χρήστης μπορεί να χρησιμοποιήσει το JWT για να στέλνει αιτήματα HTTP στον εξυπηρετητή και η συσκευή για να συνδεθεί στον MQTT Broker, ώστε να δημοσιεύει περιοδικά τα απαραίτητα μηνύματα και να λαμβάνει εντολές από τον χρήστη.



Εικόνα 21 : Διάγραμμα ακολουθίας, της επικοινωνίας του χρήστη και της συσκευής με τον διακομιστή

4.2 Το λογισμικό του μικροελεγκτή ESP-32

Το λογισμικό του μικροελεγκτή αναπτύχθηκε χρησιμοποιώντας τις γλώσσες C και C++, ενώ χρησιμοποιήθηκαν και οι κατάλληλες βιβλιοθήκες για τον έλεγχο του WiFi, του Bluetooth και για την λήψη της μέτρησης κατανάλωσης από τον μετρητή SDM120M .

Το πρώτο σημείο στο οποίο πρέπει να δοθεί σημασία, είναι η συνάρτηση αρχικοποίησης του Bluetooth Low Energy(BLE). Στην Εικόνα 22 φαίνεται ο καθορισμός των υπηρεσιών και των χαρακτηριστικών του BLE. Συγκεκριμένα, δημιουργείται μια υπηρεσία που διαθέτει δύο χαρακτηριστικά. Το πρώτο είναι τύπου “write”, δηλαδή εγγράψιμο και επιτρέπει στο χρήστη να στείλει στην συσκευή τα στοιχεία για την σύνδεση στο ασύρματο δίκτυο WiFi. Το δεύτερο χαρακτηριστικό είναι τύπου “read”, δηλαδή αναγνώσιμο και επιτρέπει στην κινητή συσκευή του χρήστη να λάβει το μοναδικό κωδικό της συσκευής.

```
BLEDevice::init("HomeIO");
BLEServer *pServer = BLEDevice::createServer();
BLEService *pService = pServer->createService(SERVICE_UUID);

BLECharacteristic *aCharacteristic = pService->createCharacteristic(
                                       A_CHARACTERISTIC_UUID,
                                       BLECharacteristic::PROPERTY_WRITE
                                     );

BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                       CHARACTERISTIC_UUID,
                                       BLECharacteristic::PROPERTY_READ
                                     );
```

Εικόνα 22 : Κώδικας αρχικοποίησης Bluetooth Low Energy

Μια ακόμα συνάρτηση του κώδικα που πρέπει να αναφερθεί είναι αυτή που πραγματοποιεί την σύνδεση της συσκευής στο ασύρματο δίκτυο WiFi και φαίνεται στην Εικόνα 23. Εφόσον η συσκευή έχει λάβει τα στοιχεία SSID και PASSWORD, μπορούν να

ξεκινήσουν οι προσπάθειες σύνδεσης. Αν η συσκευή δεν καταφέρει να συνδεθεί, θα κάνει επανεκκίνηση ώστε να επιλύσει τυχόν προβλήματα .

```
void initWiFi() {
  WiFi.mode(WIFI_STA);
  Serial.println(ssid.c_str());
  Serial.println(password.c_str());
  WiFi.begin(ssid.c_str(), password.c_str());
  Serial.print("Connecting to WiFi ..");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
    reconnectCounter++;
    resetIfNecessary();
  }
  Serial.println(WiFi.localIP());
}
```

Εικόνα 23 : Συνάρτηση αρχικοποίησης WiFi

Το βασικότερο ίσως τμήμα του κώδικα που πρέπει να αναφερθεί, είναι εκείνο που είναι υπεύθυνο για την επικοινωνία με το μετρητή μέσω του πρωτοκόλλου Modbus. Για το σκοπό αυτό χρησιμοποιήθηκε η βιβλιοθήκη SDM, η οποία καθιστά την επικοινωνία με μετρητές κατανάλωσης μέσω Modbus απλή και εύκολη. Στην Εικόνα 24, φαίνεται ο κώδικας για την αρχικοποίηση της επικοινωνίας με το μετρητή SDM120M.

```
SDM sdm(Serial1, SDM_UART_BAUD, 25, SERIAL_8N1, SDM_RX_PIN, SDM_TX_PIN);
```

Εικόνα 24 : Κώδικας αρχικοποίησης επικοινωνίας με το μετρητή κατανάλωσης

Η πρώτη παράμετρος ορίζει την σειριακή διεπαφή που θα χρησιμοποιηθεί για την επικοινωνία, η οποία είναι η Serial1. Θα μπορούσε να χρησιμοποιηθεί οποιαδήποτε από τις τρείς σειριακές διεπαφές που διαθέτει ο μικροελεγκτής εκτός από την Serial0 που χρησιμοποιείται για την εμφάνιση μηνυμάτων σχετικά με την κατάσταση της συσκευής στο εργαλείο Serial Monitor του Arduino IDE. Οι παράμετροι 2, 5 και 6 ορίζονται σε ένα αρχείο ρυθμίσεων της βιβλιοθήκης ως σταθερές και αφορούν το ρυθμό επικοινωνίας Baud (Baud Rate) το οποίο στην περίπτωση της εργασίας αυτής είναι 2400, και οι παράμετροι 5 και 6 αντιπροσωπεύουν τις επαφές 16 και 17 του μικροελεγκτή ESP-32. Η παράμετρος 3 αφορά την επαφή γενικής χρήσης (GPIO Pin), που είναι υπεύθυνη για την εναλλαγή της επικοινωνίας από αποστολή σε λήψη και το αντίστροφο. Τέλος, η παράμετρος 4 αποτελεί σταθερά του συστήματος και ορίζει την επικοινωνία σε 8 bit, χωρίς bit ισοτιμίας (parity bit) και με 1 bit τέλους (stop bit). Αφού γίνει αρχικοποίηση της επικοινωνίας, η μέτρηση των Watt μπορεί να ληφθεί με την γραμμή κώδικα που φαίνεται στην Εικόνα 25. Εδώ απαιτούνται δύο παράμετροι. Η πρώτη είναι η διεύθυνση μνήμης που θα αναγνωστεί και η δεύτερη είναι το αναγνωριστικό του μετρητή κατανάλωσης.

```
float power = sdm.readVal(0x000C, 0x01);
```

Εικόνα 25 : Κώδικας λήψης της μέτρησης κατανάλωσης

Στην συνέχεια, η μέτρηση της κατανάλωσης θα πρέπει να αποσταλεί στο διακομιστή με την χρήση του πρωτοκόλλου MQTT. Ο κώδικας αυτός μοιάζει με εκείνον στην Εικόνα 26. Το παράδειγμα αυτό είναι απλοποιημένο για λόγους αναγνωσιμότητας.

```
client.publish(deviceCode+"/instant", "{'device':'abcdef','timestamp':'2022-09-27T17:25:17.738Z','value':'25.5'}");
```

Εικόνα 26 : Αποστολή καταγραφής κατανάλωσης στο διακομιστή μέσω του πρωτοκόλλου MQTT

Το μήνυμα που αποστέλλεται περιέχει τα εξής στοιχεία. Το μοναδικό κωδικό της συσκευής, την ημερομηνία και ώρα της καταγραφής και την τιμή της κατανάλωσης. Η ημερομηνία και ώρα λαμβάνεται από το διακομιστή μέσω ενός αιτήματος HTTP με την μέθοδο GET. Ο κώδικας αυτός εμφανίζεται στην Εικόνα 27.

```
static std::string getDateTime(){
  HTTPClient http;
  std::string payload=datetime;
  http.begin(datetimeEndpoint.c_str());
  int httpResponseCode = http.GET();
  if (httpResponseCode>0) {
    Serial.print("Current DateTime: ");
    payload = std::string(http.getString().c_str());
    Serial.println(http.getString());
  }else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
  return payload;
}
```

Εικόνα 27 : Κώδικας για την λήψη της ημερομηνίας και ώρας από το διακομιστή

Παράλληλα, η συσκευή αναμένει την λήψη μηνύματος μέσω MQTT από το διακομιστή. Τα μηνύματα αυτά αφορούν την ενεργοποίηση ή απενεργοποίηση της συσκευής απομακρυσμένα από το χρήστη. Η συνάρτηση που καλείται για την ανάλυση του μηνύματος από το MQTT και την ενεργοποίηση ή απενεργοποίηση της συσκευής, εμφανίζεται στην Εικόνα 28.

```
void MQTTcallback(char* topic, byte* message, unsigned int length) {
  String messageTemp;
  for (int i = 0; i < length; i++) {
    messageTemp += (char)message[i];
  }

  JSONVar mqttObject = JSON.parse(messageTemp);
  JSONVar state = mqttObject["value"];
  if(state){
    digitalWrite(RELAY_PIN, HIGH);
    deviceState=true;
  }else{
    digitalWrite(RELAY_PIN, LOW);
    deviceState=false;
  }
}
```

Εικόνα 28 : Κώδικας για την λήψη και ανάλυση των μηνυμάτων MQTT

Τέλος, πρέπει να γίνει αναφορά στον κώδικα που επιτρέπει στο χρήστη να αλληλοεπιδρά με την συσκευή μέσω των κουμπιών της. Αρχικά, κατά την εκκίνηση της συσκευής, οι επαφές γενικής χρήσης (GPIO) 12 (SET_MODE_WIFI_PIN), 14 (SET_MODE_BLE_PIN) και 18 (RESET_PIN), ορίζονται ως είσοδοι με τον κώδικα στην Εικόνα 29 μέσα στην μέθοδο setup(). Η μέθοδος setup εκτελείται μια φορά, κατά την εκκίνηση της συσκευής.

```
pinMode(SET_MODE_WIFI_PIN, INPUT);
pinMode(SET_MODE_BLE_PIN, INPUT);
pinMode(RESET_PIN, INPUT);
```

Εικόνα 29 : Κώδικας αρχικοποίησης εισόδων κουμπιών

Στην συνέχεια, δημιουργήθηκε η μέθοδος button_press_listener(), η οποία καλείται κατά την εκτέλεση του κώδικα, ώστε να ελεγχθεί αν ο χρήστης πατάει κάποιο κουμπί. Ο κώδικας της μεθόδου αυτής, φαίνεται στην Εικόνα 30. Αν πατηθεί το κουμπί για την ενεργοποίηση του Bluetooth, τότε το WiFi θα πρέπει να απενεργοποιηθεί και αντίστροφα. Αν πατηθεί το κουμπί της επαναφοράς, η συσκευή θα κάνει επανεκκίνηση. Ελέγχεται επίσης, αν το κουμπί έμεινε πατημένο για κάποιο διάστημα, ώστε να αποφευχθούν λάθη.

```
static void button_press_listener(){
  if(digitalRead(SET_MODE_BLE_PIN) && wifiOn){
    wifiOn = false;
    Serial.println("[BLE MODE ON]");
  }
  if(digitalRead(SET_MODE_WIFI_PIN) && !wifiOn){
    wifiOn = true;
    Serial.println("[WIFI MODE ON]");
  }
  if(digitalRead(RESET_PIN) != lastResetButtonState && digitalRead(RESET_PIN)){
    lastDebounceTime = millis();
    lastResetButtonState = 1;
  }
  if(digitalRead(RESET_PIN) && (millis() - lastDebounceTime) > 50){
    Serial.println("[RESET DEVICE]");
    ESP.restart();
  }else{
    lastResetButtonState = 0;
  }
  delay(10);
}
```

Εικόνα 30 : Κώδικας για τον έλεγχο του πατήματος των κουμπιών της συσκευής

4.3 Το λογισμικό του εξυπηρετητή

Η συγκεκριμένη υποενότητα αναφέρεται στον κώδικα που εκτελείται στον εξυπηρετητή. Ο κώδικας αυτός, είναι γραμμένος στην γλώσσα Javascript και βασίζεται στο πλαίσιο εφαρμογής (framework) express.js.

Το βασικότερο κομμάτι του κώδικα, είναι η δημιουργία των RESTful web services. Τα web services είναι χωρισμένα σε κατηγορίες και οι βασικότερες είναι αυτές των χρηστών, των διαχειριστών, των συσκευών και των καταναλώσεων. Το κάθε web service ανάλογα την λειτουργία του, προστατεύεται από ένα ενδιάμεσο λογισμικό (Middleware) το οποίο επιβεβαιώνει ότι ο χρήστης είναι συνδεδεμένος στο σύστημα και διαθέτει τα κατάλληλα δικαιώματα πρόσβασης. Στην Εικόνα 31 και στην Εικόνα 32 φαίνονται οι αρχικοποιήσεις των web services και η συνάρτηση για την σύνδεση του χρήστη στο σύστημα αντίστοιχα.

```
app.use('/devices', check_if_auth, deviceRoutes);
app.use('/users', check_if_auth, userRoutes);
app.use('/consumptions', check_if_auth, consumptionRoutes);
app.use('/admin', check_if_admin, adminRoutes);
app.use('/groups', check_if_auth, groupRoutes);
```

Εικόνα 31 : Αρχικοποίηση των Web Services

Ο χρήστης στέλνει στα στοιχεία σύνδεσης του με την μέθοδο HTTP Post, και αφού ελεγχθεί η ορθότητα τους τότε, είτε δημιουργείται και επιστρέφεται ένα JWT, είτε επιστρέφεται ένα μήνυμα λάθους.

```
app.post('/users/login',
    prevent_brute_force,
    body('email').normalizeEmail().trim().isEmail().withMessage('Το email πρέπει να είναι της μορφής mail@mail.com'),
    body('password').isLength({ max: 50 }).withMessage('Ο κωδικός πρέπει να είναι 8-20 χαρακτήρες'),
    async (req, res) => {
        const errors = validationResult(req);
        console.log(errors.array());
        if (!errors.isEmpty()) {
            return res.status(400).json({ message: errors.array()[0].msg });
        }
        let email = (req.body.email).toLowerCase().trim();
        const user = await User.findOne({ email: email });
        try {
            if (user) {
                if (!user.verified) {
                    return res.status(403).json({ message: 'Δεν έχετε ενεργοποιήσει τον λογαριασμό σας.' });
                }
                const auth = await bcrypt.compare(req.body.password, user.password);
                if (auth) {
                    const token = createToken(user._id, req);
                    resetAttempts(req);
                    return res.status(200).json({ user: user, jwt: token });
                }

                logger('failed_login', null, user._id.toString(), "Αποτυχία σύνδεσης");
                failedAttempt(req);
                return res.status(403).json({ message: 'Το email και ο κωδικός δεν ταιριάζουν' });
            }
            failedAttempt(req);
            return res.status(403).json({ message: 'Δεν υπάρχει λογαριασμός με το συγκεκριμένο email' });
        } catch (err) {
            console.log(err);
            return res.status(500).json(err);
        }
    });
```

Εικόνα 32 : Κώδικας για την σύνδεση του χρήστη στο σύστημα

Με κάθε αποτυχημένη προσπάθεια σύνδεσης, ο μετρητής των αποτυχιών για την IP διεύθυνση του χρήστη αυξάνεται. Στην περίπτωση που ο χρήστης κάνει πάνω από 5 αποτυχημένες προσπάθειες να συνδεθεί, το σύστημα τον αποτρέπει από το να συνεχίσει. Αυτό γίνεται με την χρήση του ενδιάμεσου λογισμικού (middleware) 'prevent_brute_force'. Αν ο χρήστης καταφέρει να συνδεθεί σε λιγότερες από 5 προσπάθειες, οι αποτυχημένες προσπάθειες επαναφέρονται στο 0. Ο κώδικας που ελέγχει το όριο των προσπαθείων φαίνεται στην Εικόνα 33. Το όριο των προσπαθειών σύνδεσης μπορεί να οριστεί και από μια μεταβλητή περιβάλλοντος στο αρχείο .env.

```
const brute_store = {};
const MAX_ATTEMPTS = process.env.MAX_ATTEMPTS || 5;
const ATTEMPT_TIMEOUT = process.env.ATTEMPT_TIMEOUT || 24; //hours
const prevent_brute_force = (req, res, next) => {
    let now = Date.now();
    //check for block timeout
    Object.keys(brute_store).forEach((ip) => {
        if (now - brute_store[ip].last_attempt >= ATTEMPT_TIMEOUT * 60 * 60) {
            brute_store[ip] = { attempts: 0, last_attempt: 0 };
        }
    });
    let ip = req.headers['x-forwarded-for'] || req.connection.remoteAddress;
    if (brute_store.hasOwnProperty(ip) && brute_store[ip].attempts >= MAX_ATTEMPTS) {
        return res.status(429).json({ message: 'Too many requests' });
    } else {
        next();
    }
}
```

Εικόνα 33 : Κώδικας για την αποφυγή επιθέσεων brute force

Η σταθερά 'brute_store' έχει τον ρόλο μιας δομής αποθήκευσης key-value (κλειδιού-τιμής), με το κλειδί να είναι η IP διεύθυνση ενός χρήστης και την τιμή να είναι το σύνολο των προσπαθειών σύνδεσης και η στιγμή της τελευταίας προσπάθειας. Όπως φαίνεται, σε κάθε κλήση της μεθόδου 'prevent_brute_force', ελέγχεται αν κάποιος μετρητής έχει λήξει, ώστε να επανέλθει στο 0.

Σημαντικό είναι να παρατεθεί και μια αναλυτική λίστα με τις διαθέσιμες υπηρεσίες, και το πως κάποιος μπορεί να τις χρησιμοποιήσει. Αυτό σημαίνει πως οποιοσδήποτε, έχει την δυνατότητα να επεκτείνει το παρόν σύστημα με επιπλέον λειτουργίες και χαρακτηριστικά. Οι διαθέσιμες υπηρεσίες διαμορφώνονται όπως παρουσιάζονται παρακάτω.

/users/register (POST)

Εγγραφή του χρήστη στο σύστημα.

Πεδίο	Τύπος	Περιγραφή
email	Email	Το email του χρήστη
password	String	Ο κωδικός του χρήστη

/users/login (POST)
Σύνδεση του χρήστη στο σύστημα. Επιστρέφει ένα Json Web Token το οποίο ο χρήστης χρησιμοποιεί για την αυθεντικοποίηση του από το σύστημα. Συγκεκριμένα θα πρέπει να αποστέλλεται σε κάθε αίτημα του χρήστη προς τον εξυπηρετητή μέσα στο Header με την μορφή “Authorization: <Json Web Token>” .

Πεδίο	**Τύπος**	**Περιγραφή**
email	Email	Το email του χρήστη
password	String	Ο κωδικός του χρήστη

/users/update (POST)
Ενημέρωση των στοιχείων του χρήστη.

Πεδίο	**Τύπος**	**Περιγραφή**
email	Email	Το email του χρήστη
password	String	Ο κωδικός του χρήστη

/users/confirm/:token (GET)
Επιβεβαίωση του λογαριασμού του χρήστη.

Πεδίο	**Τύπος**	**Περιγραφή**
token	String	Το token που θα λάβει στο email του αμέσως μετά την εγγραφή του

/admin/uptime (GET)
Επιστροφή του χρόνου από την εκκίνηση του συστήματος σε μορφή Unix Epoch.

/admin/users (GET)
Επιστροφή όλων των χρηστών του συστήματος.

/admin/users/count (GET)
Επιστροφή του αριθμού των χρηστών του συστήματος.

/admin/devices/count (GET)
Επιστροφή του αριθμού των μοναδικών συσκευών του συστήματος.

/admin/instances (GET)
Επιστροφή του αριθμού των συνολικών εγγεγραμμένων συσκευών στο σύστημα.

/admin/logs/:limit? (POST)
Επιστροφή των καταγραφών του συστήματος. Αν το πεδίο “limit” μείνει κενό, επιστρέφει 50 καταγραφές.

Πεδίο	**Τύπος**	**Περιγραφή**
limit (προαιρετικό)	Integer (default: 50)	Ο αριθμός των εγγραφών που θα επιστρέψει

/devices (GET)
Επιστροφή όλων των συσκευών του συνδεδεμένου χρήστη, χαρακτηρίζοντας την κάθε μια ως συνδεδεμένη (online) ή αποσυνδεδεμένη (offline).

/devices/:deviceId (GET)
Επιστροφή μιας συγκεκριμένης συσκευής με βάση το id της.

Πεδίο	**Τύπος**	**Περιγραφή**
deviceId	String	Το id της συσκευής στην βάση δεδομένων

/devices/create (POST)
Εγγραφή μιας νέας συσκευής στο σύστημα.

Πεδίο	**Τύπος**	**Περιγραφή**
name	String	Όνομα συσκευής
description	String	Περιγραφή συσκευής
device_code	String	Μοναδικός κωδικός συσκευής
icon (προαιρετικό)	String	Κωδικός εικονιδίου
color (προαιρετικό)	String	Κωδικός χρώματος

/devices/update/:deviceId (POST)
Ενημέρωση των στοιχείων μιας συγκεκριμένης συσκευής με βάση το id της.

Πεδίο	**Τύπος**	**Περιγραφή**
name	String	Όνομα συσκευής
description	String	Περιγραφή συσκευής
device_code	String	Μοναδικός κωδικός συσκευής
icon (προαιρετικό)	String	Κωδικός εικονιδίου
color (προαιρετικό)	String	Κωδικός χρώματος

/devices/delete/:deviceId (POST)
Διαγραφή μιας συγκεκριμένης συσκευής με βάση το id της.

Πεδίο	**Τύπος**	**Περιγραφή**
deviceId	String	Το id της συσκευής στην βάση δεδομένων

/devices/share (POST)
Πρόσκληση πρόσβασης για την χρήση μιας συσκευής από άλλο χρήστη.

Πεδίο	**Τύπος**	**Περιγραφή**
email	Email	Το email του χρήστη που θα αποκτήσει πρόσβαση στην συσκευή
device	String	Το id της συσκευής

/devices/share/accept (POST)
Αποδοχή μιας πρόσκλησης.

Πεδίο	Τύπος	Περιγραφή
share	String	Το id της πρόσκλησης πρόσβασης

/devices/share/reject (POST)
Απόρριψη μιας πρόσκλησης.

Πεδίο	Τύπος	Περιγραφή
share	String	Το id της πρόσκλησης πρόσβασης

/devices/share/cancel (POST)
Ακύρωση μιας πρόσκλησης.

Πεδίο	Τύπος	Περιγραφή
share	String	Το id της πρόσκλησης πρόσβασης

/devices/shares (POST)
Επιστροφή των προσκλήσεων πρόσβασης που έχουν γίνει στον χρήστη.

/devices/shares/sent (POST)
Επιστροφή των προσκλήσεων πρόσβασης που έχει στείλει ο χρήστης.

/groups (GET)
Επιστροφή όλων των ομάδων συσκευών του συνδεδεμένου χρήστη.

/groups/:groupId (GET)
Επιστροφή μιας ομάδας συσκευών με βάση το id της.

Πεδίο	Τύπος	Περιγραφή
groupId	String	Το id της ομάδας συσκευών στην βάση δεδομένων

/groups/create (POST)
Εγγραφή μιας νέας συσκευής στο σύστημα.

Πεδίο	Τύπος	Περιγραφή
name	String	Όνομα ομάδας συσκευών
description	String	Περιγραφή ομάδας συσκευών

/groups/update/:groupId (POST)
Ενημέρωση των στοιχείων μιας συγκεκριμένης συσκευής με βάση το id της.

Πεδίο	Τύπος	Περιγραφή
name	String	Όνομα ομάδας συσκευών
description	String	Περιγραφή ομάδας συσκευών

/groups/delete/:groupId (POST)

Διαγραφή μιας συγκεκριμένης συσκευής με βάση το id της.

Πεδίο	Τύπος	Περιγραφή
groupId	String	Το id της ομάδας συσκευών στην βάση δεδομένων

/consumptions/raw (POST)

Επιστροφή των μετρήσεων κατανάλωσης για ένα χρονικό διάστημα, χωρίς σύμπτυξη των δεδομένων.

Πεδίο	Τύπος	Περιγραφή
from	Date	Ημ/νία εκκίνησης
to	Date	Ημ/νία τερματισμού

/consumptions/compact (POST)

Επιστροφή των μετρήσεων κατανάλωσης για ένα χρονικό διάστημα, με σύμπτυξη των δεδομένων με βάση το διάστημα που επιλέγεται. Συγκεκριμένα, αν το επιλεγμένο χρονικό διάστημα είναι μικρότερο των δύο ημερών τότε τα δεδομένα είναι με ακρίβεια ώρας, αν είναι μικρότερο των 32 ημερών τα δεδομένα είναι με ακρίβεια ημέρας, αν είναι μικρότερο των 200 ημερών είναι με ακρίβεια εβδομάδας, για διαστήματα μικρότερα των 750 ημερών είναι με ακρίβεια μήνα, ενώ για διαστήματα μεγαλύτερα των 750 ημερών είναι με ακρίβεια έτους.

Πεδίο	Τύπος	Περιγραφή
from	Date	Ημ/νία εκκίνησης
to	Date	Ημ/νία τερματισμού

/consumptions/stats (POST)

Επιστροφή των μετρήσεων κατανάλωσης για διαστήματα της τελευταίας εβδομάδας, τελευταίων 30 ημερών, τελευταίου εξαμήνου και τελευταίου έτους.

Πεδίο	Τύπος	Περιγραφή
device_instance_id	String	Το id της συσκευής του χρήστη

Επιπλέον, ένα ακόμα σημαντικό μέρος του κώδικα του εξυπηρετητή είναι τα μοντέλα των οντοτήτων του συστήματος. Πιο συγκεκριμένα, είναι αυτά που περιγράφουν την μορφή με την οποία τα δεδομένα αποθηκεύονται στην βάση δεδομένων. Ως παράδειγμα, στην Εικόνα 34 παρουσιάζεται το μοντέλο του χρήστη στο οποίο διακρίνονται τα ονόματα των πεδίων του, καθώς και ο τύπος τους. Για παράδειγμα, το πεδίο email, είναι τύπου αλφαριθμητικό, πρέπει να είναι μοναδικό και είναι απαραίτητο για την δημιουργία ενός χρήστη και το πεδίο role είναι και αυτό αλφαριθμητικό, με την διαφορά ότι δέχεται δύο συγκεκριμένες τιμές. Είτε την τιμή user, είτε την τιμή admin, με την τιμή user να είναι ο προκαθορισμένος ρόλος ενός χρήστη.

```
const UserSchema = mongoose.Schema({
    email: { type: String, unique: true, required: true},
    password: { type: String, minLength: 6, required: true },
    role: { type: String, enum: ['user', 'admin'], default: 'user' },
    verified: { type: Boolean, default: false },
    active: { type: Boolean, default: true }, verification_token: { type: String },
    last_login: { type: Date, default: Date.now },
    date: { type: Date, default: Date.now }
});
```

Εικόνα 34 : Μοντέλο χρήστη με την βιβλιοθήκη κώδικα Mongoose.js

Ένα ακόμα τμήμα κώδικα που αξίζει να αναφερθεί είναι εκείνο που είναι υπεύθυνο για την δημιουργία και την λειτουργία του MQTT Server. Στον κώδικα στην Εικόνα 35 γίνεται αρχικοποίηση του MQTT Server παίρνοντας ως παράμετρο την πόρτα ακρόασης από το αρχείο περιβάλλοντος αν αυτό υπάρχει ή την προκαθορισμένη πόρτα 1883.

```
initMQTT(process.env.MQTT_PORT || 1883);
```

Εικόνα 35 : Κλήση της συνάρτησης αρχικοποίησης MQTT Server

Ενώ, στην Εικόνα 36, φαίνεται ο κώδικας του MQTT Server που αναλαμβάνει την προώθηση των μηνυμάτων στα κατάλληλα κανάλια επικοινωνίας καθώς και την αυθεντικοποίηση των συνδεδεμένων πελατών, οι οποίοι μπορεί να είναι, είτε χρήστες του συστήματος, είτε συσκευές.

```
const initMQTT = (port) => {
    server.listen(port, function () {
        console.log('MQTT server listening on port', port)
    })
    ws.createServer({
        server: httpServer
    }, aedes.handle)
    httpServer.listen(wsPort, function () {
        console.log('WS server listening on port', wsPort)
    })
    aedes.authenticate = async function (client, username, password, callback) {
        try {
            let { auth, type } = await isAuth(password.toString());
            if (auth === null) {
                callback(null, false);
            }
            client.entity_object = auth;
            client.entity_type = type;
            callback(null, auth);
        } catch (e) {
            console.log('[MQTT ERROR] UNAUTHORIZED ENTITY');
        }
    }
    aedes.on('publish', function (packet, client) {
        console.log("[PUBLISHING]");
        if (client) {
            commandParser(packet, client); //send packet to parse
        }
    })
}
```

Εικόνα 36 : Δημιουργία MQTT Server με την βιβλιοθήκη κώδικα Aedes.js

Κατά την εκκίνηση του MQTT, ουσιαστικά δημιουργούνται δύο servers. Ένας με το πρωτόκολλο του MQTT και ένας με το πρωτόκολλο WebSocket. Αυτό εξυπηρετεί στην συνδεσιμότητα του συστήματος, με διαφορετικές τεχνολογίες. Στο παρόν σύστημα, η κινητή εφαρμογή χρησιμοποιεί το πρωτόκολλο WebSocket, ενώ η συσκευή χρησιμοποιεί το πρωτόκολλο MQTT. Επίσης, στην διαδικασία της αυθεντικοποίησης της κάθε οντότητας, το σύστημα ορίζει αν πρόκειται για συσκευή ή για χρήστη, ώστε να επιτρέπει την πρόσβαση στα κατάλληλα κανάλια επικοινωνίας.

Για να συνδεθεί κάποιος στο MQTT Server, θα πρέπει πρώτα να γίνει αυθεντικοποίηση του χρήστη ώστε να λάβει το απαραίτητο Json Web Token (JWT). Στην συνέχεια, το JWT μπορεί να χρησιμοποιηθεί κατά την σύνδεση ενός MQTT client στον server αποστέλλοντας το στο πεδίο password όπως φαίνεται και στο παράδειγμα στην Εικόνα 37.

```
mqtt.connect({servers : [{ host: 'localhost', port: 1883}], username : '', password :token});
```

Εικόνα 37 : Παράδειγμα κώδικα για την σύνδεση στον MQTT Server

Αφού επιτευχθεί η σύνδεση με τον server, ο χρήστης μπορεί να χρησιμοποιήσει συγκεκριμένα κανάλια-θέματα (topics) για να επικοινωνήσει με τις συσκευές του. Τα κανάλια αυτά είναι τα εξής:

- **ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ/state** : Έλεγχος της συσκευής για το άνοιγμα και κλείσιμο του κυκλώματος. Πρέπει να σταλεί μήνυμα τύπου String και στην μορφή JSON " { device: 'ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ', value: 'true ή false' } ".
- **ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ/consumption** : Στο κανάλι αυτό, η συσκευή δημοσιεύει κάθε λεπτό μια καταγραφή κατανάλωσης kWh. Το μήνυμα είναι τύπου String και στην μορφή JSON "{ device: 'ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ', timestamp: 'HM/NIA ΚΑΙ ΩΡΑ ΚΑΤΑΓΡΑΦΗΣ', value: 'ΚΑΤΑΝΑΛΩΣΗ ΣΕ kWh' }".
- **ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ/instant** : Στο κανάλι αυτό, η συσκευή δημοσιεύει κάθε μερικά δευτερόλεπτα μια καταγραφή κατανάλωσης σε Watt. Το μήνυμα είναι τύπου String και στην μορφή JSON "{ device: 'ΚΩΔΙΚΟΣ_ΣΥΣΚΕΥΗΣ', timestamp: 'HM/NIA ΚΑΙ ΩΡΑ ΚΑΤΑΓΡΑΦΗΣ', value: 'ΚΑΤΑΝΑΛΩΣΗ ΣΕ Watt }".

Κατά την σύνδεση μιας συσκευής στο σύστημα, εκπέμπεται το γεγονός 'client' το οποίο ενημερώνει την κατάσταση της συσκευής σε 'online', καλώντας την μέθοδο 'updateClientStates' με παραμέτρους το αναγνωριστικό της συσκευής και την λογική τιμή true. Ο κώδικας της λειτουργίας αυτής φαίνεται στην Εικόνα 38.

```
aedes.on('client', function (client, err) {
    console.log('[CLIENT CONNECTED]', client.id)
    if (client.entity_type == "device") {
        updateClientState(client.id, true);
    }
})
```

Εικόνα 38 : Κώδικας ακρόασης γεγονότος (event listener) σύνδεσης συσκευής

Αν μια συσκευή αποσυνδεθεί από το MQTT για περισσότερα από 30 δευτερόλεπτα, εκπέμπεται το γεγονός 'clientDisconnect' όπως φαίνεται στον κώδικα στην Εικόνα 39. Έπειτα, το σύστημα αλλάζει την κατάσταση της σε 'offline', καλώντας την μέθοδο 'updateClientStates' με παραμέτρους το αναγνωριστικό της συσκευής και την λογική τιμή false.

```
aedes.on('clientDisconnect', function (client, err) {
    console.log('[CLIENT DISCONNECTED]', client.id)
    if (client.entity_type == "device") {
        updateClientState(client.id, false);
    }
})
```

Εικόνα 39 : Κώδικας ακρόασης γεγονότος (event listener) αποσύνδεσης συσκευής

Έτσι, ο χρήστης μπορεί να ελέγξει αν μια συσκευή του αντιμετωπίζει πρόβλημα, και να προχωρήσει στις απαραίτητες ενέργειες (π.χ. έλεγχος της σύνδεσης στο διαδίκτυο). Το τμήμα του λογισμικού που φροντίζει για την ενημέρωση της κατάστασης των συσκευών, φαίνεται στην Εικόνα 40. Η κατάσταση των συσκευών αποθηκεύεται σε μια δομή τύπου κλειδιού-τιμής (key-value) στην μεταβλητή 'allClients', με το κλειδί να είναι το αναγνωριστικό (id) της συσκευής.

```
const allClients = {};
const updateClientState = async (clientId, state) => {
    allClients[clientId] = state;
    console.log(allClients[clientId] ? '[CLIENT ONLINE]' : '[CLIENT OFFLINE]', clientId);
}
```

Εικόνα 40 : Αποθήκευση κατάστασης συσκευής σε δομή key-value

Γενικότερα, σε ότι αφορά την ασφάλεια του συστήματος, τηρήθηκαν όλες οι διαδικασίες και οι βέλτιστες πρακτικές. Η λίστα των βέλτιστων πρακτικών υπάρχει διαθέσιμη στην σελίδα τεκμηρίωσης (documentation) του πλαισίου εφαρμογών Express.js.[9] Η επικοινωνία με τον διακομιστή, σε κάθε περίπτωση, γίνεται κρυπτογραφημένα, μέσω του πρωτοκόλλου κρυπτογράφησης TLS. Για να διασφαλιστεί αυτό, όλα τα μη ασφαλή αιτήματα HTTP ανακατευθύνονται σε HTTPS, μέσω του κώδικα που φαίνεται στην Εικόνα 41.

```
app.enable('trust proxy');
app.use(function (request, response, next) {
    if (!request.secure) {
        return response.redirect("https://" + request.headers.host + request.url);
    }
    next();
});
```

Εικόνα 41 : Ανακατεύθυνση των αιτημάτων HTTP σε HTTPS

[9] https://expressjs.com/en/advanced/best-practice-security.html

4.4 Η εφαρμογή για έξυπνες κινητές συσκευές

Στην υποενότητα αυτή, γίνεται λόγος για την κινητή εφαρμογή. Συγκεκριμένα αναφέρονται οι λειτουργίες στις οποίες έχει πρόσβαση ο χρήστης, αλλά και τα σημαντικότερα τμήματα του κώδικα της. Στα πλαίσια αυτής της διπλωματικής εργασίας, δημιουργήθηκε εκτελέσιμη εφαρμογή για συσκευές με έκδοση Android 4.3 και πάνω. Ωστόσο, ο κώδικας που αναπτύχθηκε παρέχει την δυνατότητα και για παραγωγή εκτελέσιμου αρχείου για συσκευές IOS.

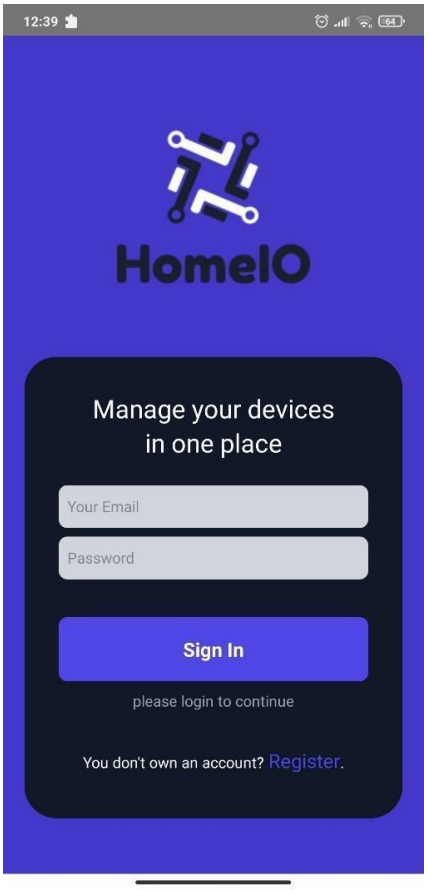
4.4.1 Λειτουργικότητα εφαρμογής

Η βασικότερη ίσως προϋπόθεση για την επιτυχία οποιασδήποτε εφαρμογής, είναι η φιλικότητα της προς το μέσο χρήστη. Αν μια εφαρμογή χαρακτηρίζεται από περίπλοκη εμφάνιση ή δυσλειτουργίες, καθίσταται δύσκολη στην χρήση της άρα και αποτυγχάνει. Έτσι οι χρήστες στρέφονται προς άλλες λύσεις και παρόμοιες εφαρμογές της αγοράς. Για το λόγο αυτό, η λογική που ακολουθήθηκε κατά το σχεδιασμό των διεπαφών είναι η απλότητα και η συμμετρικότητα.

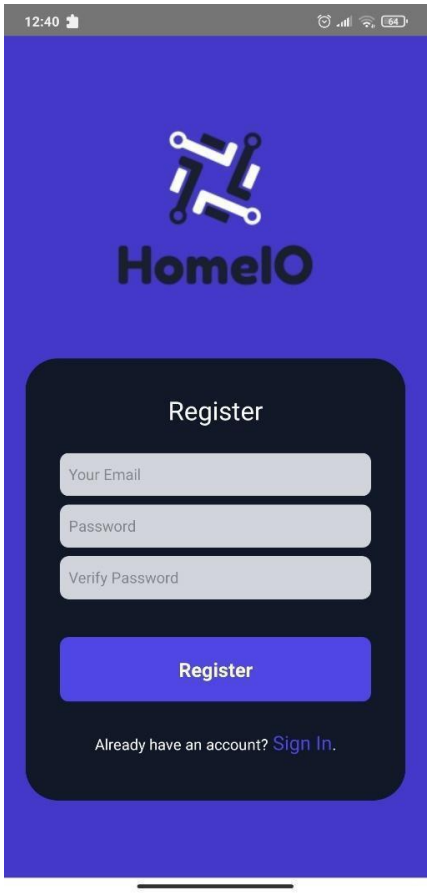
Πριν παρουσιαστούν και αναλυθούν οι διεπαφές του χρήστη πρέπει να σημειωθεί ότι τα δεδομένα που εισάγονται επικυρώνονται τόσο στην ίδια την εφαρμογή όσο και στο λογισμικό του εξυπηρετητή όπου και αποστέλλονται. Σε περίπτωση που τα δεδομένα που εισάγονται δεν είναι έγκυρα, εμφανίζεται ανάλογο μήνυμα σφάλματος. Τέτοια μηνύματα σφάλματος μπορεί να αφορούν διπλότυπη εγγραφή στο σύστημα, μη έγκυρη εισαγωγή κωδικού, μη έγκυρη εισαγωγή email ή κάποιο γενικό σφάλμα π.χ. εξαιτίας αδυναμίας πρόσβασης στο διαδίκτυο, αδυναμία επικοινωνίας με τον εξυπηρετητή κλπ.

Αρχικές οθόνες μη συνδεδεμένου χρήστη

Οι αρχικές οθόνες της εφαρμογής για τον χρήστη πριν αυτός συνδεθεί στο σύστημα είναι αυτές της σύνδεσης (Εικόνα 42) και της εγγραφής (Εικόνα 43) στο σύστημα. Για να συνεχίσει ο χρήστης, θα πρέπει να εισάγει τα στοιχεία σύνδεσης του ή να κάνει εγγραφή στο σύστημα.



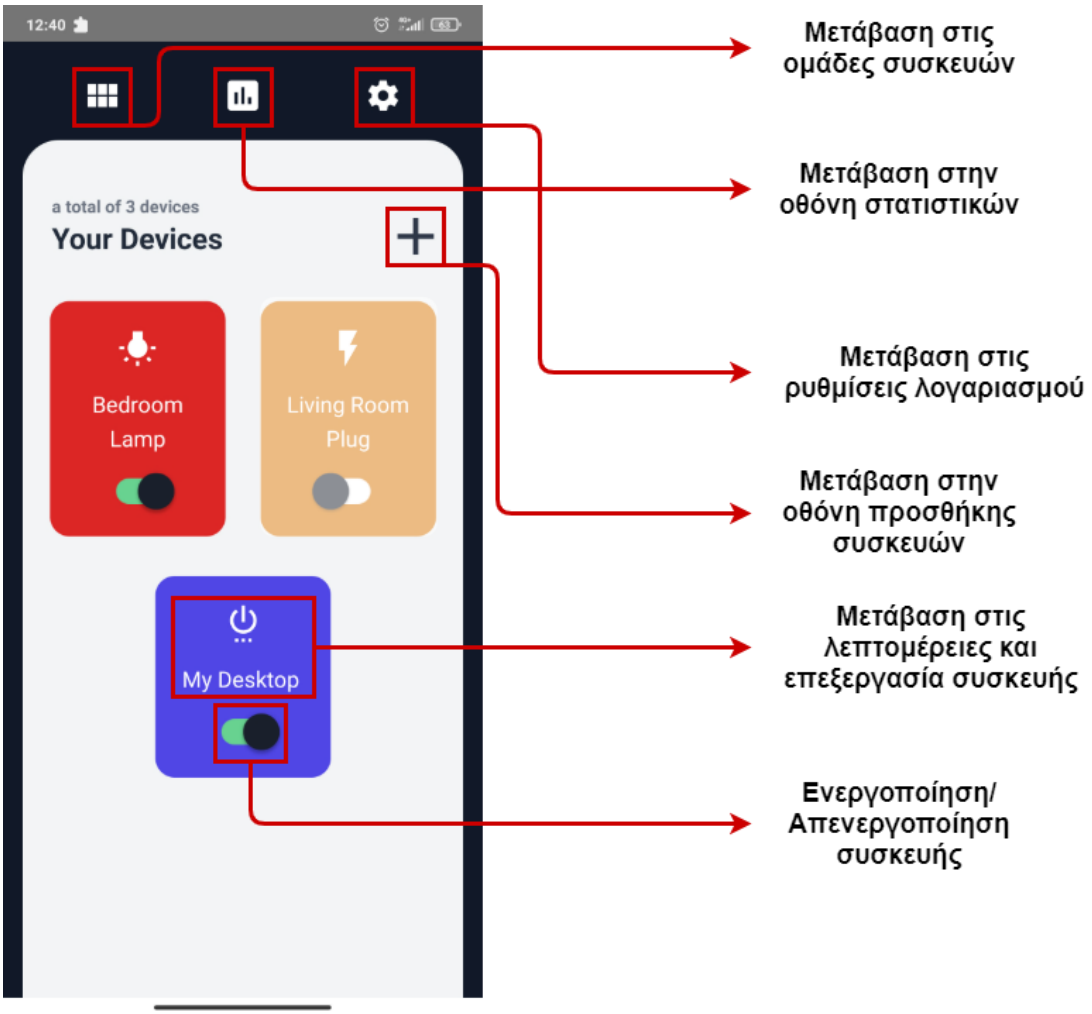
Εικόνα 42 : Οθόνη σύνδεσης



Εικόνα 43 : Οθόνη εγγραφής

<u>Αρχική οθόνη συνδεδεμένου χρήστη</u>

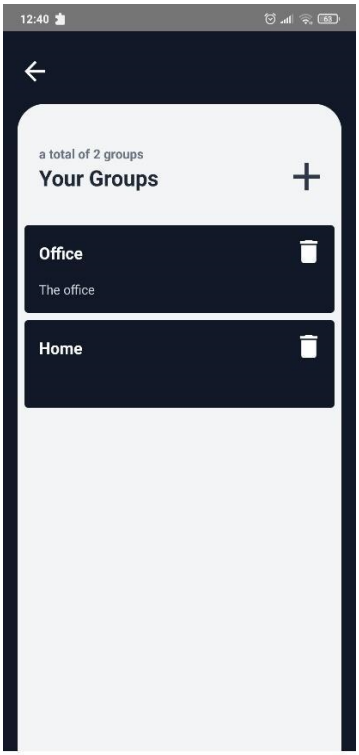
Η οθόνη που βλέπει ο χρήστης μετά την σύνδεση του στην εφαρμογή, είναι αυτή μέσω της οποίας έχει πρόσβαση στην ενεργοποίηση ή απενεργοποίηση των συσκευών του, αλλά και στις υπόλοιπες λειτουργίες του συστήματος. Από εδώ, ο χρήστης μπορεί να κατευθυνθεί στην οθόνη των ομάδων συσκευών, στα στατιστικά, στην οθόνη προσθήκης και επεξεργασίας συσκευών, καθώς και στην οθόνη ρυθμίσεων του λογαριασμού του. Ένα στιγμιότυπο της οθόνης με επεξήγηση του μενού και των επιλογών, φαίνεται στην Εικόνα 44. Στο στιγμιότυπο αυτό, η συσκευή 'Living Room Plug' εμφανίζεται «θαμπή» καθώς είναι αποσυνδεδεμένη.



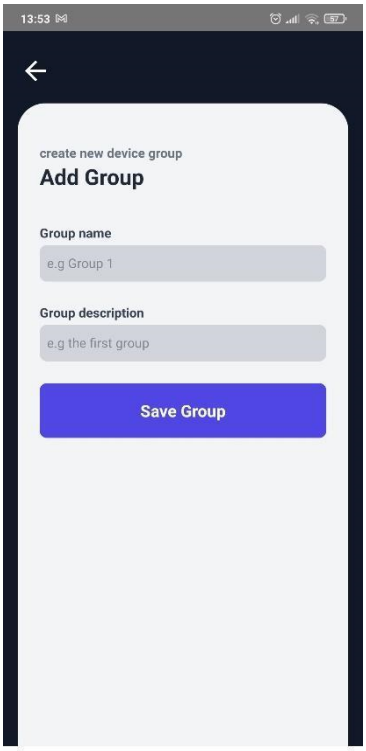
Εικόνα 44 : Αρχική οθόνη συνδεδεμένου χρήστη

Οθόνη ομάδων συσκευών

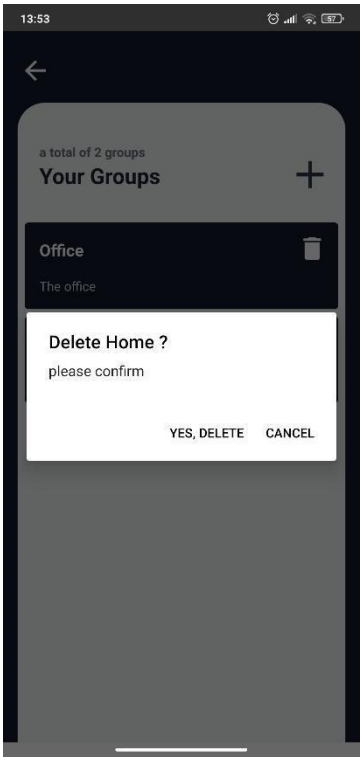
Στην οθόνη αυτή, ο χρήστης μπορεί να δει τις ομάδες συσκευών του, οι οποίες στην ουσία χρησιμεύουν στον διαχωρισμό των συσκευών σε υποκατηγορίες για την καλύτερα διαχείριση τους. Επίσης, ο χρήστης της εφαρμογής εδώ μπορεί να προσθέσει νέα ομάδα ή να διαγράψει κάποια από τις υπάρχουσες. Τα στιγμιότυπα των οθονών που σχετίζονται με τις ομάδες συσκευών, φαίνονται στην Εικόνα 45, την Εικόνα 46 και την Εικόνα 47.



Εικόνα 45 : Οθόνη ομάδων συσκευών



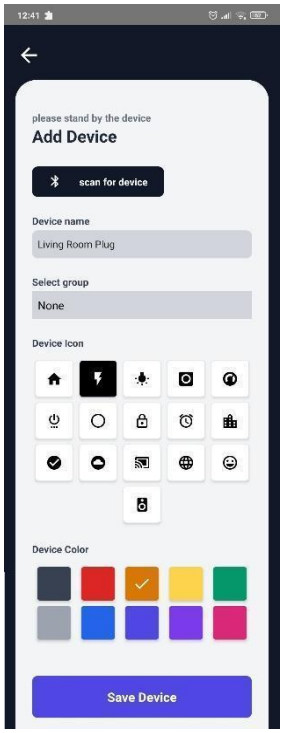
Εικόνα 46 : Οθόνη προσθήκης ομάδας συσκευών



Εικόνα 47 : Διαγραφή ομάδας συσκευών

Οθόνη προσθήκης συσκευών

Στην οθόνη αυτή, ο χρήστης μπορεί να προσθέσει στον λογαριασμό του μια νέα συσκευή. Πριν το κάνει αυτό, θα πρέπει να συνδεθεί με την συσκευή μέσω Bluetooth ώστε να της κοινοποιήσει το όνομα και τον κωδικό του ασύρματου δικτύου, να επιλέξει ένα όνομα, ένα χρώμα και ένα εικονίδιο. Προαιρετικά, μπορεί να την κατηγοριοποιήσει σε κάποια ομάδα συσκευών. Η οθόνη για την προσθήκη νέας συσκευής φαίνεται στην Εικόνα 48. Πατώντας το κουμπί του Bluetooth, ο χρήστης καλείται να εισάγει το SSID και τον κωδικού του τοπικού δικτύου WiFi. Προϋπόθεση είναι, η ενεργοποίηση του Bluetooth της κινητής συσκευής.



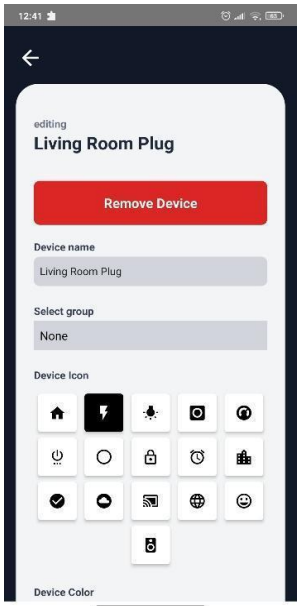
Εικόνα 48 : Οθόνη προσθήκης συσκευής

Οθόνη στατιστικών συσκευής

Εικόνα 49 : Οθόνη στατιστικών συσκευής

Στην οθόνη αυτή, ο χρήστης μπορεί να δει μερικά στατιστικά στοιχεία για την κατανάλωση της συσκευής σε συγκεκριμένα χρονικά διαστήματα, όπως επίσης και την τρέχουσα χρήση ισχύος σε Watt. Ένα στιγμιότυπο της οθόνης αυτής είναι η Εικόνα 49.

Οθόνη επεξεργασίας συσκευών



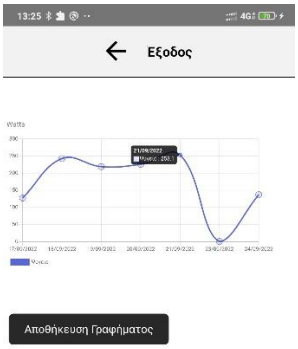
Εικόνα 50 : Οθόνη επεξεργασίας συσκευής

Η οθόνη αυτή είναι πρακτικά ίδια με την οθόνη προσθήκης συσκευών, με τη μόνη διαφορά στην εμφάνιση να είναι το κουμπί διαγραφής της συσκευής. Εδώ ο χρήστης μπορεί να επεξεργαστεί το όνομα της συσκευής του, να αλλάξει την ομάδα της, το εικονίδιο της και το χρώμα της. Στην Εικόνα 50, παρουσιάζεται ένα στιγμιότυπο επεξεργασίας μιας συσκευής.

Οθόνη γραφημάτων

Εδώ, ο χρήστης της εφαρμογής έχει στην διάθεση του δύο φίλτρα, όπως φαίνεται και στην Εικόνα 51.Ένα για το χρονικό εύρος για το οποίο θέλει να εμφανίσει γραφήματα και ένα για το ποιες συσκευές θέλει να συμπεριλάβει. Πατώντας προβολή γραφημάτων, ο χρήστης μεταβαίνει στην οθόνη που εμφανίζεται στην Εικόνα 52. Εκεί έχει την επιλογή να αποθηκεύσει το διάγραμμα στην συσκευή του σε μορφή εικόνας JPEG.

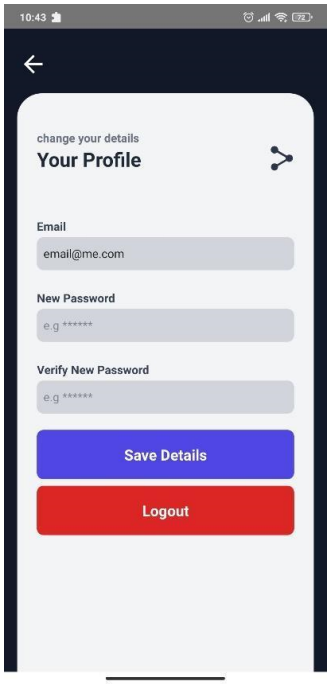
Εικόνα 51 : Οθόνη δημιουργίας γραφημάτων



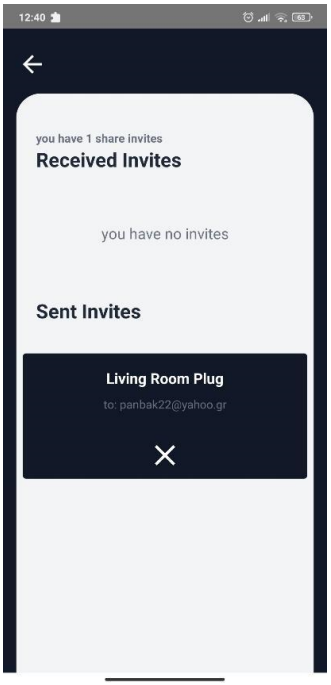
Εικόνα 52 : Οθόνη γραφημάτων

Οθόνη λογαριασμού χρήστη

Σε αυτό το σημείο της εφαρμογής, ο χρήστης έχει πρόσβαση στα δεδομένα του λογαριασμού του. Ειδικότερα, έχει την δυνατότητα να αλλάξει το email και τον κωδικό του λογαριασμού του. Επιπλέον, πατώντας το εικονίδιο διαμοιρασμού το οποίο βρίσκεται δίπλα στον τίτλο της οθόνης, ο χρήστης μεταβαίνει στις προσκλήσεις που έχει στείλει ή δεχθεί για πρόσβαση σε κάποια συσκευή. Εκεί, μπορεί να αποδεχθεί κάποια πρόσκληση, να την απορρίψει ή να ακυρώσει μια πρόσκληση που έστειλε ο ίδιος. Στιγμιότυπα των λειτουργειών αυτών φαίνονται στην Εικόνα 53 και Εικόνα 54.



Εικόνα 53 : Οθόνη επεξεργασίας στοιχείων χρήστη



Εικόνα 54 : Οθόνη κοινής χρήσης συσκευών

4.4.2 Κώδικας εφαρμογής

Εδώ παρουσιάζονται τα βασικά τμήματα του κώδικα στον οποίο βασίζεται η κινητή εφαρμογή. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Javascript και το πλαίσιο εφαρμογής React Native. Επίσης, χρησιμοποιήθηκαν κάποιες native βιβλιοθήκες κώδικα για IOS και Android.

Τα θεμελιώδη στοιχεία που συνδυάζει το React Native για την δημιουργία πολύπλοκων μοντέρνων εφαρμογών είναι:

- Τα components
- Η γλώσσα σήμανσης JSX
- Τα props
- Το state

Τα components είναι επαναχρησιμοποιούμενα τμήματα κώδικα τα οποία περιέχουν την εμφάνιση και την λειτουργικότητα ενός τμήματος της εφαρμογής. Σε συνδυασμό με την γλώσσα JSX, το React Native δίνει την δυνατότητα δημιουργίας γραφικού περιβάλλοντος με τρόπο που θυμίζει πολύ HTML. Ένα component δέχεται κάποιες ιδιότητες (props) οι οποίες είναι οι παράμετροι που μπορεί να χρησιμοποιήσει και του παρέχονται κατά την αρχικοποίηση του. Τέλος, έχουμε τις ειδικές μεταβλητές της εφαρμογής (state), των οποίων όταν η τιμή αλλάζει, προκαλούν και τις κατάλληλες αλλαγές στην οθόνη.Ένα παράδειγμα φαίνεται στην Εικόνα 55.

```
const DeviceCard = (props) => {
    const [isEnabled, setIsEnabled] = useState(props.isEnabled);

    const toggleSwitch = () => {
        setIsEnabled(props.onToggle());
    }

    return (
        <View style={[tailwind(`${props.color} w-5/12 mb-8 px-3 py-4 items-center flex rounded-2xl`)]}>
            <TouchableOpacity
                style={[tailwind(`items-center flex`)]}
                onPress={props.onPress}
            >
                <Icon color="white"
                    name={props.icon}
                    height="30" width="30"
                    style={[tailwind('mt-2')]}
                />
                <Text style={[tailwind('text-white text-lg my-4 text-center')]}>{props.name}</Text>
            </TouchableOpacity>
            <Switch
                trackColor={{ false: "#fff", true: "#68d391" }}
                thumbColor={"#1a202c"}
                ios_backgroundColor="#fff"
                style={[tailwind('mb-2'), { transform: [{ scaleX: 1.5 }, { scaleY: 1.5 }] }]}
                onValueChange={toggleSwitch}
                value={isEnabled}
            />
        </View>
    );
};
```

Εικόνα 55 : Παράδειγμα κώδικα component στο React Native

Βασική λειτουργία για το σύστημα που υλοποιήθηκε, είναι η δυνατότητα αποστολής εντολών για την ενεργοποίηση/απενεργοποίηση μιας συσκευής μέσω του πρωτοκόλλου MQTT. Στο στιγμιότυπο στην Εικόνα 56, φαίνεται η μέθοδος που αναλαμβάνει την αλλαγή της κατάστασης μιας συσκευής. Για την υλοποίηση του MQTT client, χρησιμοποιήθηκε το πακέτο react native MQTT.

```
const toggleDeviceState = (device_code, isEnabled, index) => {
    devices[index].device.state = !isEnabled
    try {
        let packet = {
            device: device_code,
            timestamp: new Date(),
            value: !isEnabled
        };
        client.publish(device_code + '/state', JSON.stringify(packet));
    } catch (err) {
        errorAlert(err);
    }
    return (devices[index].device.state);
}
```

Εικόνα 56 : Κώδικας για την ενεργοποίηση/απενεργοποίηση συσκευής μέσω MQTT

Τέλος, κάτι ακόμα που είναι απαραίτητο να αναφερθεί, είναι η επικοινωνία της εφαρμογής με την συσκευή, μέσω Bluetooth Low Energy. Η εφαρμογή αναλαμβάνει να κοινοποιήσει στην συσκευή τα στοιχεία σύνδεσης του τοπικού ασύρματου δικτύου και να διαβάσει τον μοναδικό κωδικό της συσκευής, για την εγγραφή της στον λογαριασμό του χρήστη. Στο κομμάτι κώδικα στην Εικόνα 57, φαίνεται το κουμπί με το οποίο ενεργοποιείται η διαδικασία αποστολής των στοιχείων σύνδεσης του Wi-Fi. Το πακέτο λογισμικού που χρησιμοποιήθηκε για την υλοποίηση των λειτουργιών Bluetooth είναι το React Native BLE PLX.

```
_BleManager.connectToDevice(device.id).then(async device => {
    await device.discoverAllServicesAndCharacteristics();
    _BleManager.stopDeviceScan();
    device.readCharacteristicForService(
        "6f581fe2-b873-4fd5-b27f-ead4948a45db",
        "d0b779be-34ee-4920-9f14-c2f3e88cddca",
        null
    ).then(characteristic => {
        console.log(base64.decode(characteristic.value));
        setDeviceCode(base64.decode(characteristic.value).toLowerCase());
        setReadSuccess(true);
    });

    device.writeCharacteristicWithResponseForService(
        "6f581fe2-b873-4fd5-b27f-ead4948a45db",
        "5afe1153-5662-44f5-8783-cd706d60ddc6",
        base64.encode(wifiConnectionString)
    ).then(characteristic => {
        console.log(characteristic);
        setWriteSuccess(true);
    });
```

Εικόνα 57 : Κώδικας για την αποστολή των στοιχείων σύνδεσης του δίκτυο WiFi στην συσκευή

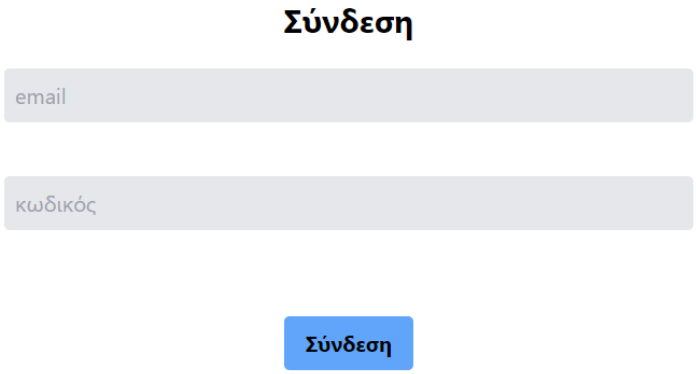
4.5 Η διαδικτυακή εφαρμογή διαχειριστή

Στην υποενότητα αυτή, αναλύεται η διαδικτυακή εφαρμογή του διαχειριστή του συστήματος. Ειδικότερα, αναφέρονται οι λειτουργίες και οι δυνατότητες που παρέχονται και στην συνέχεια παρουσιάζονται τα πιο αξιοσημείωτα τμήματα του κώδικα.

4.5.1 Λειτουργικότητα εφαρμογής διαχειριστή

Οθόνη σύνδεσης

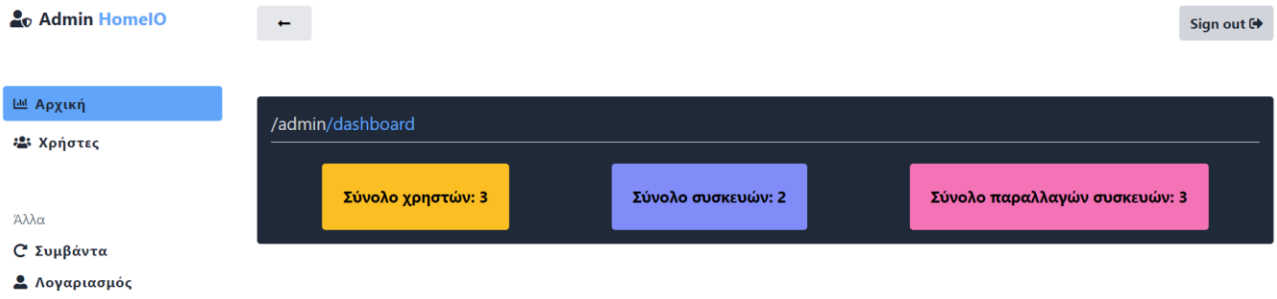
Αυτή είναι η οθόνη, μέσω της οποίας ο διαχειριστής καλείται να συνδεθεί στο σύστημα χρησιμοποιώντας το email και τον κωδικό του. Το στιγμιότυπο της, φαίνεται στην Εικόνα 58.



Εικόνα 58 : Οθόνη σύνδεσης στην εφαρμογή διαχειριστή

Οθόνη ελέγχου και στατιστικών

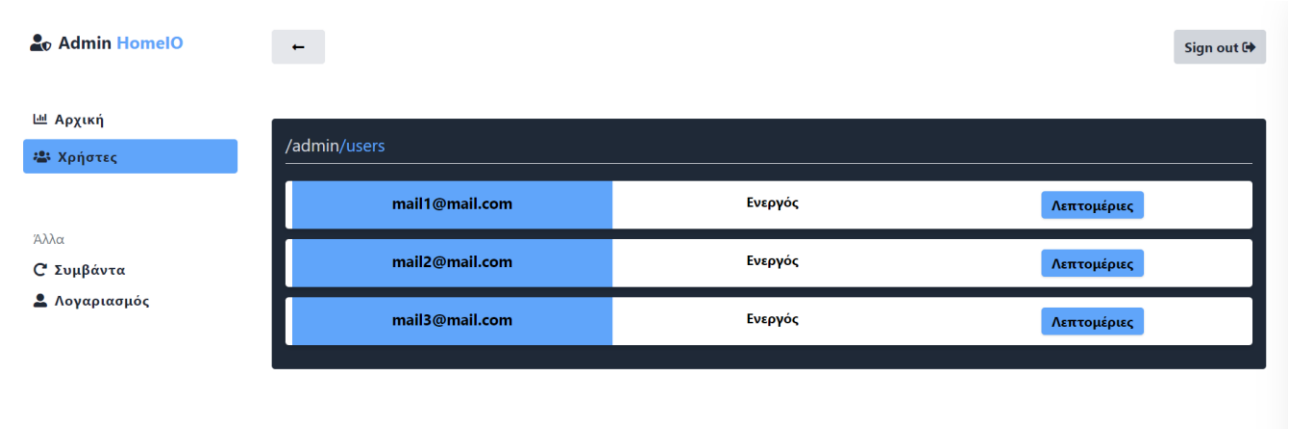
Σε αυτή την οθόνη, η οποία παρουσιάζεται στην Εικόνα 59, ο διαχειριστής του συστήματος μπορεί να δει κάποια γενικά στοιχεία σχετικά με το σύστημα. Αυτά είναι ο αριθμός των χρηστών, ο αριθμός των μοναδικών συσκευών, αλλά και το σύνολο των εγγεγραμμένων συσκευών, συμπεριλαμβανομένων των διπλότυπων (μια συσκευή μπορεί να ανήκει σε πολλαπλούς χρήστες).



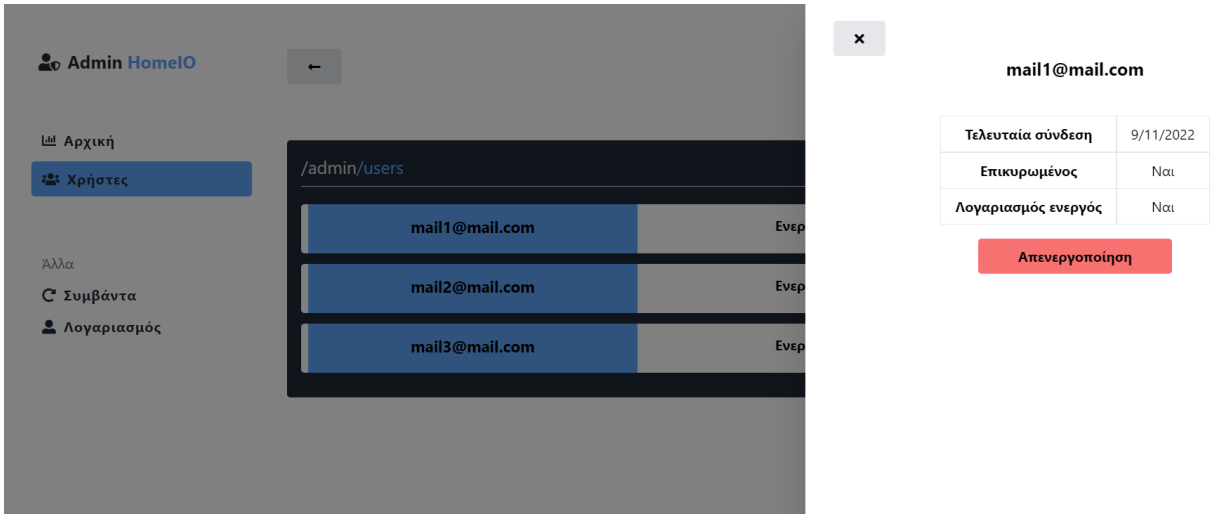
Εικόνα 59 : Οθόνη στατιστικών στην εφαρμογή διαχειριστή

Οθόνη χρηστών

Στην οθόνη αυτή, ο διαχειριστής έχει την δυνατότητα να δει το email του χρήστη, την ημερομηνία εγγραφής του, αν έχει επιβεβαιώσει το email του και αν ο λογαριασμός του είναι ενεργός. Επίσης, μπορεί να ενεργοποιήσει ή να απενεργοποιήσει τον λογαριασμό κάποιου χρήστη και να ενημερωθεί για την τελευταία φορά που αυτός συνδέθηκε πατώντας στην επιλογή “λεπτομέριες”. Τα ανάλογα στιγμιότυπα, φαίνονται στην Εικόνα 60 και στην Εικόνα 61.



Εικόνα 60 : Οθόνη χρηστών στην εφαρμογή διαχειριστή



Εικόνα 61 : Οθόνη λεπτομερειών χρήστη στην εφαρμογή διαχειριστή

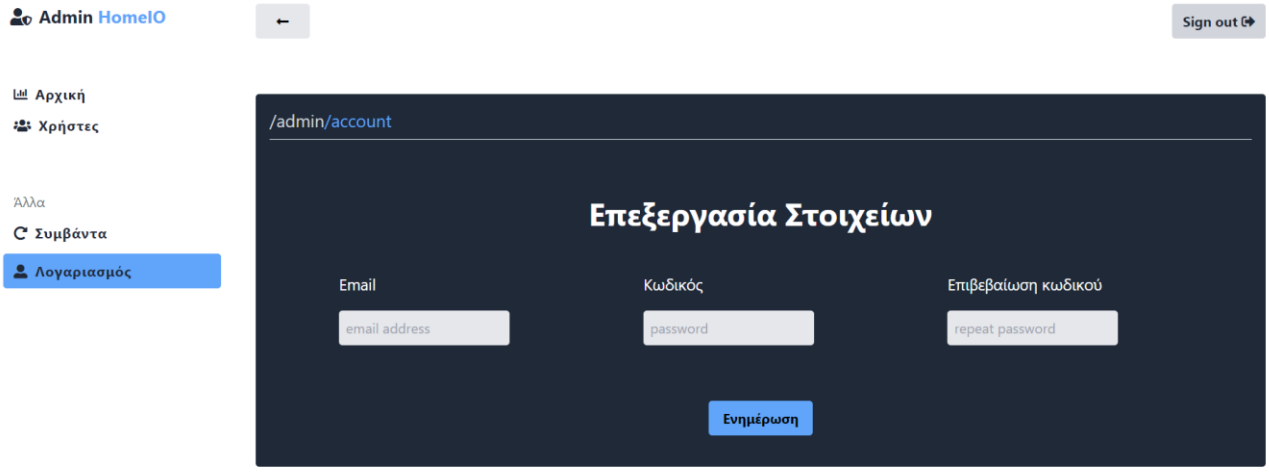
Οθόνη συμβάντων

Σε αυτή την οθόνη, ο διαχειριστής μπορεί να ενημερωθεί σχετικά με σημαντικά συμβάντα του συστήματος, όπως είναι η ενεργοποίηση/απενεργοποίηση μιας συσκευής από έναν χρήστη ή οι πολλαπλές αποτυχημένες προσπάθειες σύνδεσης σε έναν λογαριασμό.Ένα στιγμιότυπο της οθόνης συμβάντων φαίνεται στην Εικόνα 62.

Εικόνα 62 : Οθόνη συμβάντων στην εφαρμογή διαχειριστή

Οθόνη λογαριασμού

Σε αυτή την οθόνη, η οποία εμφανίζεται και στην Εικόνα 63, ο διαχειριστής μπορεί να επεξεργαστεί τα στοιχεία του λογαριασμού του, μπορεί δηλαδή να αλλάξει το email του και τον κωδικό του.



Εικόνα 63 : Οθόνη επεξεργασίας στοιχείων λογαριασμού στην εφαρμογή διαχειριστή

4.5.2 Κώδικας εφαρμογής διαχειριστή

Σε αυτή την υποενότητα, παρουσιάζονται τα βασικότερα και σημαντικότερα τμήματα του κώδικα που βρίσκεται πίσω από την διαδικτυακή εφαρμογή του διαχειριστή. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε, είναι η Javascript και το πλαίσιο εφαρμογής είναι το React.js. Τα βασικά στοιχεία που διαθέτει το React είναι ίδια με εκείνα του React Native.

Ένα βασικό τμήμα της εφαρμογής, είναι η διαδικασία της σύνδεσης του διαχειριστή στο σύστημα. Αφού ο διαχειριστής συμπληρώσει τα στοιχεία του, αυτά αποστέλλονται στον εξυπηρετητή, ώστε να πραγματοποιηθεί η αυθεντικοποίηση. Η μέθοδος που είναι υπεύθυνη για την σύνδεση του διαχειριστή στο σύστημα φαίνεται στην Εικόνα 64.

```
const login = async () => {
    let auth = await axios.post('https://home-io-server.herokuapp.com/users/login', {
        email: email,
        password: password
    }).then((response) => {
        console.log(response.data);
        if (response.status !== 200) {
            response.json().then(function (data) {
                setError(true);
            });
            return false;
        } else {
            window.localStorage.setItem('user', JSON.stringify(response.data));
            console.log(window.localStorage.getItem('user'));
            setError(false);
            return true;
        }
    }).catch((err) => {
        setError(true);
    });
    if (auth) {
        setError(false);
        window.location.reload();
    }
}
```

Εικόνα 64 : Κώδικα για την σύνδεση του χρήστη στην εφαρμογή διαχειριστή

Αν η σύνδεση στο σύστημα είναι επιτυχής, το τμήμα εκείνο που είναι υπεύθυνο για την προβολή της κατάλληλης οθόνης, μεταφέρει τον χρήστη από την οθόνη της σύνδεσης, στην οθόνη διαχείρισης. Ο κώδικας αυτός φαίνεται στην Εικόνα 65.

```
import { useState, useEffect } from 'react';
import App from './App';
import Login from './Login';
import authenticate from './authenticate';

const Authenticator = () => {
    const [isAuth, setAuth] = useState(false);
    useEffect(() => {
        authenticate().then((res) => {
            setAuth(res);
        })
    });
    return (
        <>
            {
                isAuth ? <App /> : <Login />
            }
        </>
    );
}
export default Authenticator;
```

Εικόνα 65 : Κώδικας για την αλλαγή της οθόνης μετά από επιτυχημένη σύνδεση

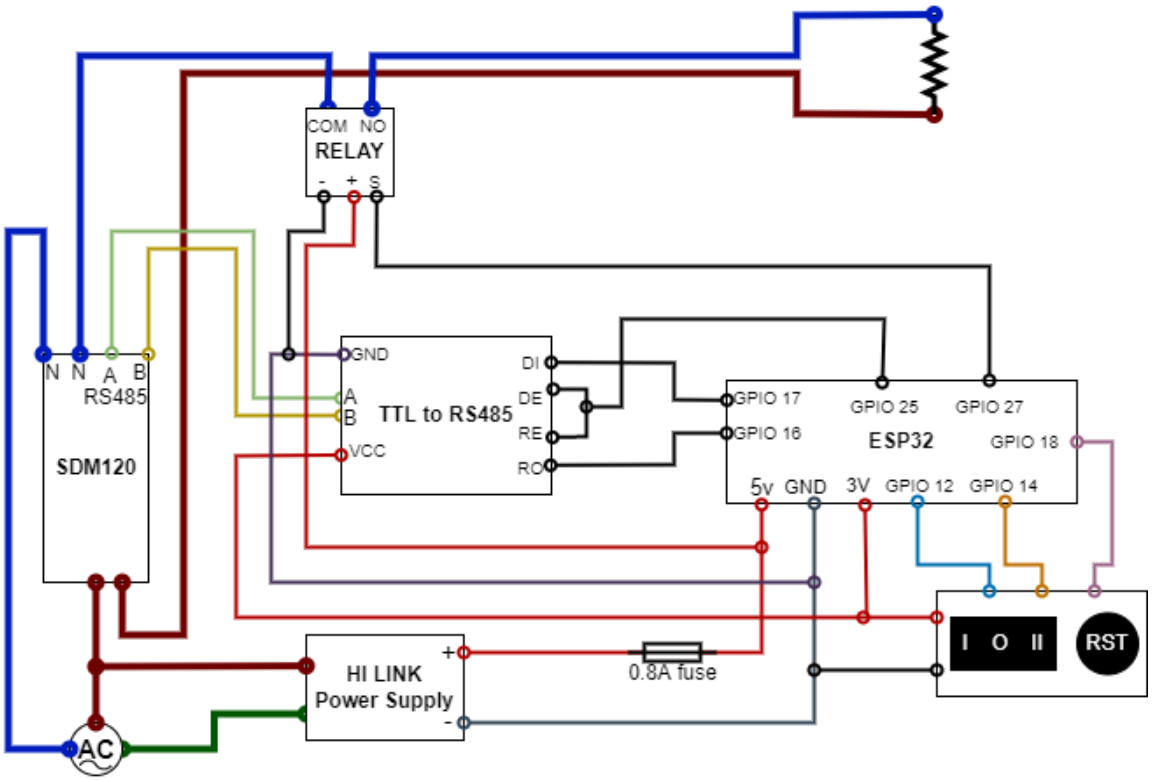
Στην περίπτωση που ο χρήστης έχει συνδεθεί πρόσφατα στο σύστημα από την ίδια συσκευή, αποστέλλεται το αποθηκευμένο JSON Web Token (JWT) και ο διαχειριστής δεν χρειάζεται να εισάγει τα στοιχεία σύνδεσης του. Ο κώδικας που στέλνει στον διακομιστή το JWT, φαίνεται στην Εικόνα 66.

```
const authenticate = async () => {
    if (!window.localStorage.getItem('user')) {
        return false;
    }
    let jwt = JSON.parse(window.localStorage.getItem('user')).jwt;
    let auth = await fetch(`${config.SERVER_BASE_URL}/authorize/`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `${jwt}`
        }
    }).then((response) => {
        if (response.status !== 200) {
            return false;
        }
        return response.json().then(function (data) {
            if (data === 1) {
                return true;
            }
            return false;
        });
    });
    return auth;
}
```

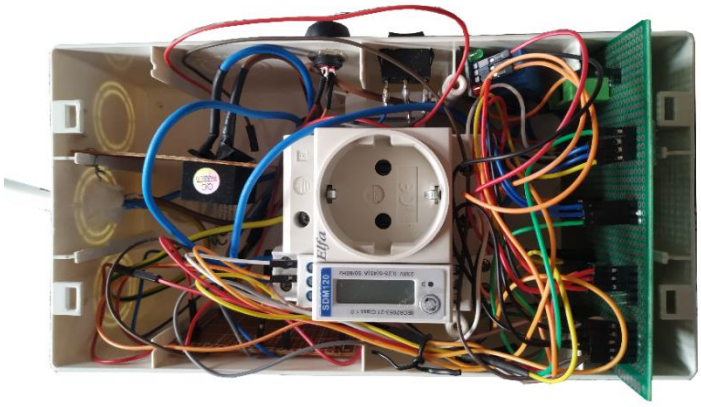
Εικόνα 66 : Κώδικας για την σύνδεση του διαχειριστή με την χρήση JWT

4.6 Η κατασκευή της συσκευής

Στην ενότητα αυτή, αναλύεται η κατασκευή της συσκευής σε ότι αφορά τα δομικά της στοιχεία, την σύνδεση τους με το μικροελεγκτή, αλλά επίσης αναλύεται εκτενώς ο τρόπος που αυτά συνδέονται και αλληλοεπιδρούν μεταξύ τους. Η συνδεσμολογία της συσκευής παρουσιάζεται στην Εικόνα 67. Στην Εικόνα 68, είναι μια φωτογραφία της συσκευής χωρίς κάλυμμα, ώστε να φαίνονται οι συνδέσεις. Τα καλώδια που χρησιμοποιήθηκαν για την σύνδεση στο ηλεκτρικό δίκτυο των 230 Volt είναι τύπου AWG14(2.5mm^2), ενώ για τις τάσεις των 3.3 και 5 Volt χρησιμοποιήθηκαν καλώδια τύπου AWG26(0.14mm^2).



Εικόνα 67 : Συνδεσμολογία της συσκευής



Εικόνα 68 : Υλοποιημένη συσκευή, χωρίς κάλυμμα

4.6.1 Η βάση της συσκευής

Ως βάση της συσκευής, αποφασίστηκε να χρησιμοποιηθεί ένα ηλεκτρολογικό κουτί. Συγκεκριμένα, τα διάφορα υποσυστήματα της συσκευής περιέχονται σε ένα ηλεκτρολογικό κουτί της εταιρείας IDE, το οποίο διαθέτει ενσωματωμένη ράγα τεσσάρων θέσεων. Για την εργασία αυτή, χρησιμοποιήθηκαν οι τρείς από τις τέσσερεις θέσης. Δύο για μια πρίζα ράγας τύπου schuko (σούκο), και μία θέση για το μετρητή κατανάλωσης SDM120M. Εκτός από την ράγα τεσσάρων θέσεων, η οποία ήταν απαραίτητη για την προσάρτηση των παραπάνω, το κουτί διαθέτει αρκετό χώρο, ώστε να ενσωματωθούν τα διάφορα ηλεκτρονικά στοιχεία και περιφερειακά κυκλώματα. Το κουτί της συσκευής, μαζί με την πρίζα και το μετρητή φαίνονται στην Εικόνα 69.

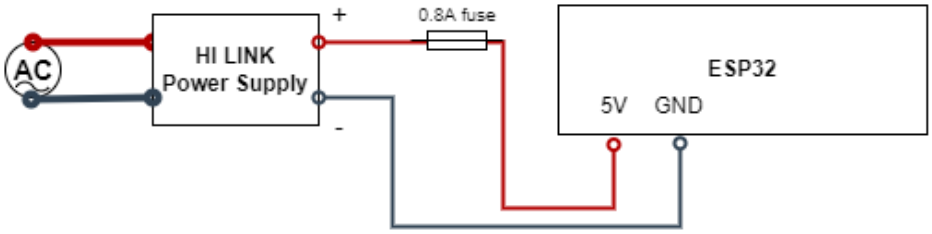
Εικόνα 69 : Φωτογραφία της συσκευής που υλοποιήθηκε, από εμπρόσθια όψη

Ωστόσο, το κουτί αυτό από μόνο του δεν ήταν αρκετό για την υλοποίηση της συσκευής, αλλά χρειάστηκαν μερικές τροποποιήσεις. Πιο συγκεκριμένα, δημιουργήθηκαν δύο τρύπες στο πλαστικό πλαίσιο. Μια στρογγυλή, για την θέση του διακόπτη επανεκκίνησης, και μια ορθογώνια, για την θέση του διακόπτη εναλλαγής των λειτουργιών Bluetooth και WiFi, όπως φαίνεται στην Εικόνα 70. Επιπλέον, έγιναν μικρές τρύπες στο εσωτερικό του κουτιού, για την στήριξη των κυκλωμάτων.

Εικόνα 70 : Φωτογραφία της συσκευής που υλοποιήθηκε, από πλάγια όψη

4.6.2 Τροφοδοσία του μικροελεγκτή ESP32

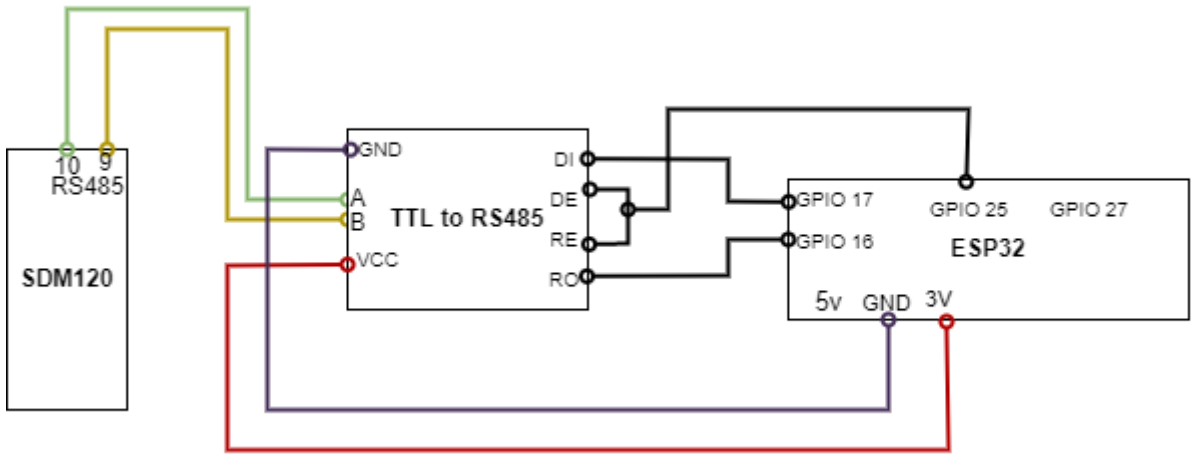
Όπως προαναφέρθηκε, η τροφοδοσία του μικροελεγκτή, αλλά και των υπόλοιπων κυκλωμάτων γίνεται με το τροφοδοτικό Hi-Link του οποίου η έξοδος είναι 5 volt. Έτσι, για την λειτουργία του μικροελεγκτή θα πρέπει να συνδεθούν ο θετικός πόλος του τροφοδοτικού με την επαφή 5v του μικροελεγκτή και ο αρνητικός πόλος με την επαφή GND (γείωση). Ως μέτρο προστασίας της συσκευής, χρησιμοποιήθηκε μια ασφάλεια με τιμή 0.8 Ampere. Η συνδεσμολογία που περιγράφεται, φαίνεται στην Εικόνα 71.



Εικόνα 71 : Συνδεσμολογία για την τροφοδοσία του ESP-32 με το τροφοδοτικό Hi-Link

4.6.3 Σύνδεση με το μετρητή κατανάλωσης

Για την σύνδεση του μικροελεγκτή με το μετρητή κατανάλωσης, χρειάστηκε ένα επιπλέον υποσύστημα, ο μετατροπέας MAX485 για την επικοινωνία της σειριακής διεπαφής του ESP32 με την διεπαφή RS485 του μετρητή κατανάλωσης. Για την χρήση του μετατροπέα, απαιτείται εκτός από την τροφοδοσία, να συνδεθούν στο μικροελεγκτή οι επαφές λήψης (RX) και μετάδοσης (TX) καθώς και μια επαφή για την εναλλαγή λειτουργίας του μετατροπέα σε λήψη και μετάδοση. Στην συνέχεια, οι επαφές A και B του μετατροπέα, συνδέονται στις επαφές 10 και 9 του μετρητή κατανάλωσης αντίστοιχα. Οι 10 και 9 είναι οι τερματικές επαφές του μετρητή για την επικοινωνία με το πρωτόκολλο Modbus. Ο τρόπος σύνδεσης παρουσιάζεται και στην Εικόνα 72.

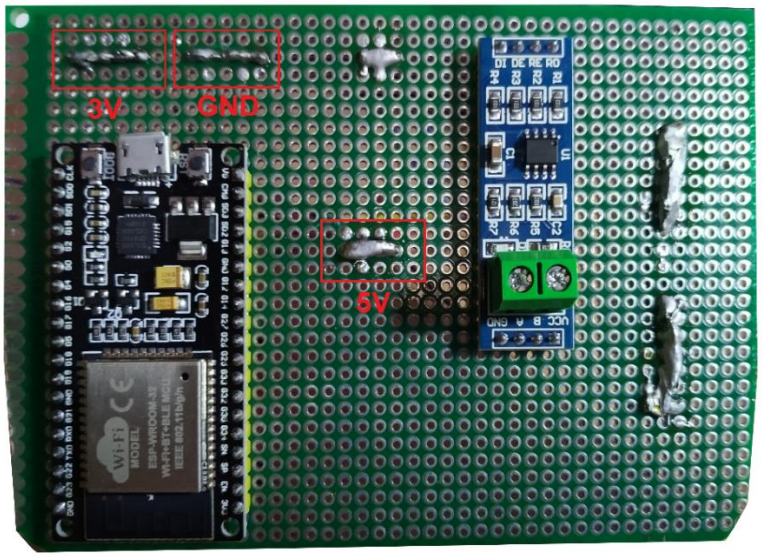


Εικόνα 72 : Συνδεσμολογία του ESP-32 με το μετρητή κατανάλωσης SDM120M

Η σειριακή διεπαφή του μικροελεγκτή, που χρησιμοποιείται για την επικοινωνία με το μετρητή κατανάλωσης, είναι η Serial1 η οποία αντιστοιχεί στις επαφές 16 (RX) και 17 (TX). Οι επαφές ελέγχου DE και RE συνδέονται μεταξύ τους και στην συνέχεια με την επαφή 25 (GPIO 25) του ESP32. Αντί για την επαφή 25, μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις επαφές εισόδου-εξόδου γενικής χρήσεως.

Στην Εικόνα 73, φαίνεται το βασικότερο κύκλωμα της συσκευής. Είναι μια διάτρητη πλακέτα στην οποία έχουν στηριχθεί ο μικροελεγκτής ESP-32, ο μετατροπέας TTL σε RS485, ενώ επίσης υπάρχει και μια ασφάλεια 800mA. Για την τροφοδοσία των

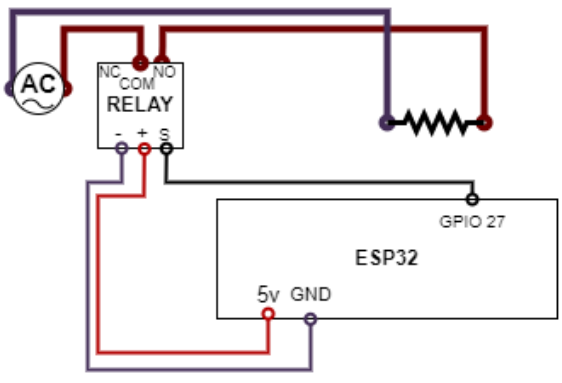
διάφορων υποσυστημάτων, έχουν κολληθεί και μερικές επιπλέον επαφές, όπως επισημαίνεται και στην φωτογραφία.



Εικόνα 73 : Διάτρητη πλακέτα στήριξης των ESP-32 και TTL σε RS485

4.6.4 Σύνδεση με το Ρελέ

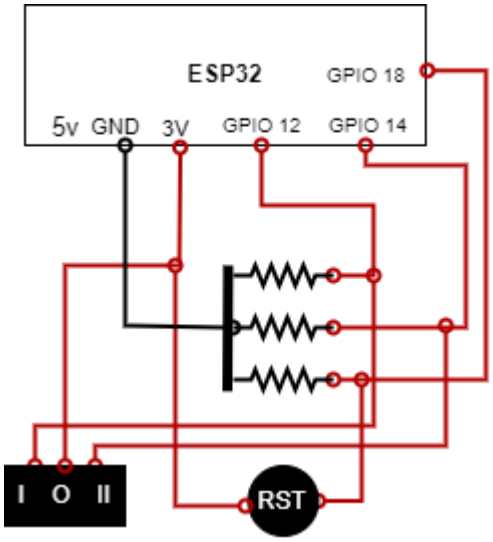
Η λειτουργία του απομακρυσμένου ελέγχου της συσκευής στηρίζεται στο ρελέ. Το ρελέ συνδέεται τόσο με το μικροελεγκτή, όσο και με το δίκτυο ηλεκτροδότησης. Συγκεκριμένα, το ρελέ διαθέτει 6 επαφές, εκ των οποίων οι πέντε από αυτές είναι συνδεδεμένες ταυτόχρονα. Οι τρεις επαφές σχετίζονται με τον έλεγχο του από το μικροελεγκτή, ενώ οι άλλες τρεις συνδέονται με την φάση του ηλεκτρικού δικτύου. Η συνδεσμολογία του ρελέ φαίνεται στην Εικόνα 74. Οι ενδείξεις NO (Normally Open) Και NC (Normally Closed) σημαίνουν ότι το κύκλωμα είναι ανοικτό ή κλειστό αντίστοιχα όταν το ρελέ είναι ανενεργό. Στην επαφή COM συνδέεται η φάση του ηλεκτρικού δικτύου. Το ρελέ τροφοδοτείται με 5 Volt και για τον έλεγχο του συνδέεται στην επαφή εισόδου-εξόδου γενικής χρήσης 27 (GPIO 27), ωστόσο θα μπορούσε να χρησιμοποιηθεί οποιαδήποτε άλλη επαφή GPIO.



Εικόνα 74 : Συνδεσμολογία ESP-32 με το Ρελέ

4.6.5 Σύνδεση διακοπτών

Η συσκευή διαθέτει δύο διακόπτες. Ένα διακόπτη τριών καταστάσεων, οποίος χρησιμεύει στην εναλλαγή των λειτουργιών WiFi και Bluetooth Low Energy, και ένα διακόπτη πίεσης για την επανεκκίνηση της συσκευής. Το σχηματικό των διακοπτών, εμφανίζεται στην Εικόνα 75.



Εικόνα 75 : Σχηματικό σύνδεσης διακοπτών με τον μικροελεγκτή ESP-32

Ο διακόπτης εναλλαγής λειτουργιών χρησιμοποιεί τις επαφές 12 και 14, ενώ ο διακόπτης πίεσης, την επαφή 18. Αντί για αυτές, θα μπορούσαν φυσικά να χρησιμοποιηθούν οποιεσδήποτε επαφές GPIO. Όπως φαίνεται στο κύκλωμα, σε κάθε επαφή υπάρχει συνδεδεμένη μια αντίσταση. Οι αντιστάσεις έχουν την τιμή 10 kΩ. Η χρησιμότητα τους έγκειται στο γεγονός ότι, οι επαφές GPIO είναι σε αβέβαιη κατάσταση. Η κατάσταση αυτή, ονομάζεται floating (επιπλέων), στην οποία άλλες φορές επικρατεί το λογικό 1 και άλλες το λογικό 0. Η διάταξη στην Εικόνα 75 ονομάζεται pull-down αντίσταση και επιλύει ακριβώς αυτό το πρόβλημα, επιτρέποντας μόνο το λογικό 0, όταν ο διακόπτης είναι ανοικτός ή το λογικό 1, όταν ο διακόπτης είναι κλειστός.

4.7 Ασφάλεια συστήματος

Στην ενότητα αυτή, αναφέρονται τα μέτρα που πάρθηκαν, με σκοπό την διαφύλαξη της ασφάλειας του συστήματος, τόσο στο κομμάτι της κατασκευής για την προστασία του χρήστη, όσο και στο κομμάτι του λογισμικού. Γίνεται λοιπόν λόγος για τα χαρακτηριστικά του συστήματος, που είναι υπεύθυνα για την ασφάλεια των χρηστών του και την προστασία τους από δυσλειτουργίες και κακόβουλες ενέργειες.

Ασφάλεια συσκευής

Για την φυσική ασφάλεια της συσκευής, λήφθηκε μέριμνα, ώστε οι συνδέσεις των καλωδίων που συνδέονται στο ηλεκτρικό δίκτυο να είναι σταθερές και μονωμένες. Επίσης, το κουτί της συσκευής περικλείει όλα τα επικίνδυνα για το χρήστη τμήματα, αφήνοντας εκτεθειμένη μόνο την πρίζα τύπου σούκο. Τα καλώδια που χρησιμοποιήθηκαν έχουν το απαραίτητο πάχος για την τάση που τα διατρέχει. Τέλος, στο κύκλωμα προστέθηκε μια ασφάλεια 800 mA για την προστασία από βραχυκυκλώματα.

Ασφάλεια λογισμικού

Για την ασφάλεια του συστήματος από κακόβουλους χρήστες έγιναν οι παρακάτω ενέργειες:

- Δημιουργήθηκαν ενδιάμεσα λογισμικά (middleware), για τον έλεγχο των δικαιωμάτων του χρήστη σε κάθε του ενέργεια. Τα middleware αυτά είναι δύο και αφορούν την διασφάλιση ότι ο χρήστης είναι συνδεδεμένος και την διασφάλιση ότι ο χρήστης είναι διαχειριστής.
- Γίνεται έλεγχος του δικαιώματος μιας συσκευής για αποστολή μετρήσεων σε ένα κανάλι. Συγκεκριμένα, ελέγχεται αν το κανάλι δημοσίευσης ανήκει στη συνδεδεμένη συσκευή.
- Γίνεται έλεγχος του δικαιώματος ενός χρήστη για την αποστολή εντολών σε ένα κανάλι. Πρέπει δηλαδή, να υπάρχει συσχέτιση του χρήστη και της συσκευής στην βάση δεδομένων για την αποστολή εντολών στα κανάλια της.
- Δημιουργήθηκε κώδικας προστασίας από επιθέσεις ωμής δύναμης (brute force), για τις ενέργειες της σύνδεσης του χρήστη και της συσκευής. Στην περίπτωση που υπάρξουν πολλαπλές αποτυχημένες προσπάθειες σύνδεσης, το σύστημα αποτρέπει την συνέχιση των προσπαθειών με βάση την διεύθυνση IP.
- Έγινε χρήση των κατάλληλων κεφαλίδων (headers), για την μετρίαση του κινδύνου από διάφορες κακόβουλες επιθέσεις, με την χρήση της βιβλιοθήκης λογισμικού helmet[10]. Το helmet, είναι μια συλλογή ενδιάμεσων λογισμικών (middleware), τα οποία πραγματοποιούν ρυθμίσεις στις κεφαλίδες αυτόματα, χωρίς να είναι απαραίτητη η παρέμβαση από τον προγραμματιστή του συστήματος.
- Απενεργοποιήθηκε η κεφαλίδα ΄x-powered-by΄ για την μείωση του fingerprinting (αναγνωρισιμότητα συστήματος). Η κεφαλίδα αυτή, παρέχει στον πιθανό κακόβουλο χρήστη πληροφορίες για το διακομιστή, γεγονός επικίνδυνο.
- Γίνεται καταγραφή των ενεργειών των χρηστών, σε θέματα σύνδεσης στο σύστημα και αποστολής εντολών στις συσκευές, με σκοπό την αναγνώριση πιθανών κακόβουλων ενεργειών.
- Η επικοινωνία των πελατών (clients) με τον διακομιστή (server), γίνεται κρυπτογραφημένα μέσω TLS (Transport Layer Security). Έτσι, η υποκλοπή δεδομένων με τεχνικές packet sniffing και Man in the Middle είναι δύσκολη έως αδύνατη.

4.8 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό, αναλύθηκαν και επεξηγήθηκαν σε βάθος τόσο οι πτυχές του λογισμικού, όσο και του υλικού μέρους της εργασίας. Έγινε αναφορά σε σημαντικά τμήματα του κώδικα ο οποίος εκτελείται στο διακομιστή, την εφαρμογή του διαχειριστή, την κινητή εφαρμογή των χρηστών, αλλά και στην συσκευή. Επιπλέον, έγινε εμβάθυνση στις λειτουργίες των εφαρμογών μέσα από αναλυτικά στιγμιότυπα, όπως και στην συνδεσμολογία των υλικών για την υλοποίηση της συσκευής στην τελική της μορφή.

[10] https://helmetjs.github.io/

Κεφάλαιο 5: Πειραματική Επαλήθευση

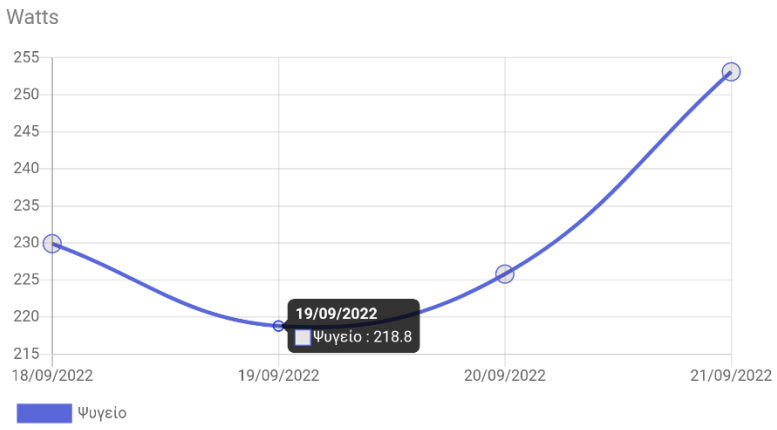
Η εξασφάλιση της ορθής και απρόσκοπτης λειτουργίας, αποτελεί βασικό μέρος στο σχεδιασμό και την υλοποίηση ενός συστήματος. Σε αυτό το κεφάλαιο, θα ελεγχθεί η ακρίβεια του υλοποιημένου συστήματος, μέσω κάποιον πραγματικών σεναρίων χρήσης. Το κάθε σενάριο συνοδεύεται από τις απαραίτητες εικόνες ή φωτογραφίες, ενώ τέλος παρατίθενται και ο σχολιασμός των αποτελεσμάτων.

5.1 Πειράματα

Πείραμα 1 – έλεγχος συνεχούς λειτουργίας

Στο πείραμα αυτό, δημιουργήθηκε ένα σενάριο, κατά το οποίο το υλοποιημένο σύστημα τέθηκε σε λειτουργία για μερικές μέρες. Συγκεκριμένα, για 4 συνεχόμενες ημέρες ο διακομιστής και η συσκευή επικοινωνούσαν για την λήψη και αποθήκευση των μετρήσεων κατανάλωσης ενός ψυγείου.

Το σύστημα αντεπεξήλθε χωρίς πρόβλημα στην συνεχή λειτουργία, και το γράφημα των μετρήσεων φαίνεται στην Εικόνα 76.



Εικόνα 76 : Γράφημα κατανάλωσης ψυγείου

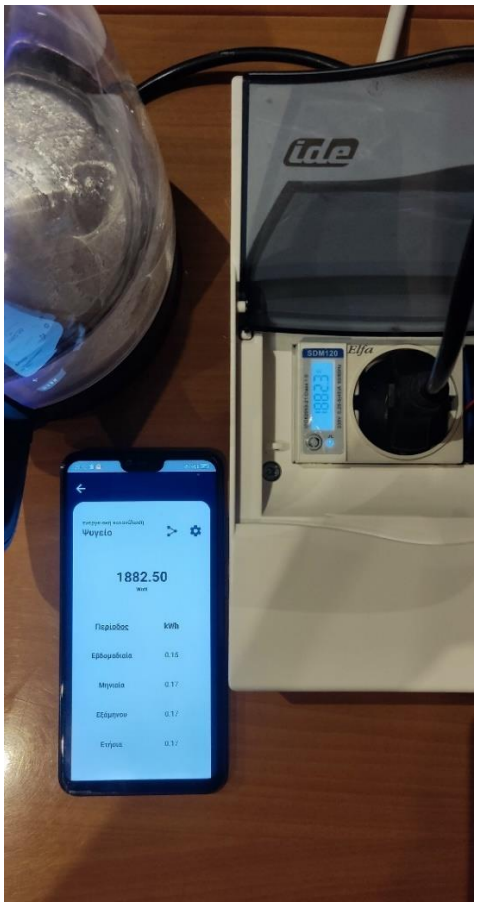
Πείραμα 2 – έλεγχος ακρίβειας μετρήσεων

Ο σκοπός αυτού του πειράματος, είναι η εξακρίβωση της ορθότητας των μετρήσεων κατανάλωσης που λαμβάνει το σύστημα. Ως συσκευές ελέγχου χρησιμοποιήθηκαν ένας βραστήρας νερού και μια ηλεκτρική σκούπα.

Ο βραστήρας αναφέρει κατανάλωση 1850-2200 Watt, όπως φαίνεται από την φωτογραφία της ετικέτας στην Εικόνα 77.

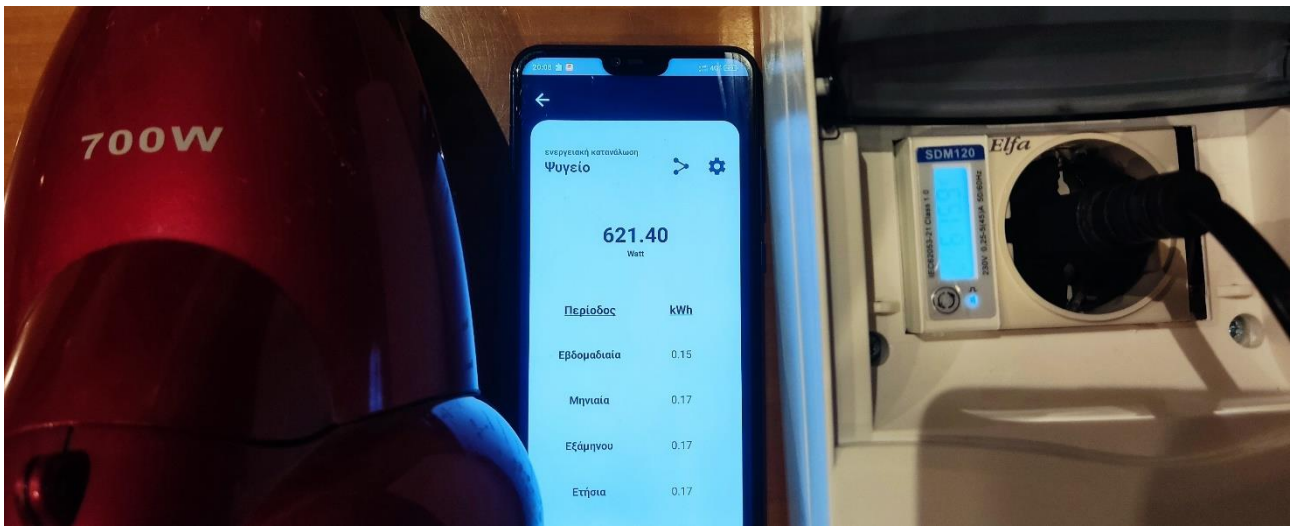
Εικόνα 77 : Φωτογραφία ετικέτας χαρακτηριστικών βραστήρα

Η κατανάλωση που μετρήθηκε είναι περίπου 1882 Watt, όπως φαίνεται και στην Εικόνα 78. Όπως διαπιστώνεται, οι μετρήσεις του συστήματος αντιστοιχούν στην αναφερόμενη κατανάλωση του βραστήρα.



Εικόνα 78 : Πείραμα μέτρησης κατανάλωσης βραστήρα

Η ηλεκτρική σκούπα αναφέρει κατανάλωση 700 Watt, όπως εμφανίζεται και στην φωτογραφία του πειράματος στην Εικόνα 79. Η μέτρηση που έλαβε το σύστημα είναι περίπου 621 Watt, το οποίο ανταπεξέρχεται στην πραγματικότητα, καθώς η απόκλιση είναι σχετικά μικρή. Η μέτρηση που λήφθηκε, φαίνεται επίσης στην Εικόνα 79 στην οθόνη της κινητής συσκευής.



Εικόνα 79 : Πείραμα μέτρησης κατανάλωσης ηλεκτρικής σκούπας

5.2 Συμπέρασμα

Από την πειραματική διαδικασία λοιπόν, προκύπτει η αξιοπιστία του συστήματος τόσο σε θέματα απρόσκοπτης λειτουργίας, όσο και σε θέματα ακρίβειας. Ακόμη διαπιστώθηκε, πως σε περίπτωση αποσύνδεσης από το διαδίκτυο η υλοποιημένη συσκευή είναι σε θέση να συνδεθεί ξανά και να συνεχίσει την αποστολή μετρήσεων.

5.3 Σύνοψη κεφαλαίου

Στο πέμπτο κεφάλαιο του έργου, διεξήχθη μια σειρά πειραμάτων με σκοπό την διαπίστωση της ορθής λειτουργίας του συστήματος. Τα πειράματα ήταν τρία στον αριθμό, και χωρίστηκαν σε δύο σκέλη. Εκείνο της συνεχούς λειτουργίας και εκείνο της ακρίβειας των μετρήσεων. Τα αποτελέσματα της πειραματικής διαδικασίας ήταν θετικά, με το σύστημα να λειτουργεί με τον αναμενόμενο τρόπο.

Κεφάλαιο 6: Επίλογος

6.1 Ανακεφαλαίωση

Το IoT σύστημα που σχεδιάστηκε και υλοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας, έχει να κάνει με τον απομακρυσμένο έλεγχο της παροχής ρεύματος σε μια συσκευή και την μέτρηση της κατανάλωσης της. Στο επίκεντρο της συσκευής IoT βρίσκεται ο μικροελεγκτής ESP-32, ενώ ο έλεγχος της από το χρήστη γίνεται χρησιμοποιώντας μια κινητή εφαρμογή. Η υλοποίηση, έγινε με γνώμονα την ευκολία στην χρήση καθώς και την ασφάλεια του χρήστη και των προσωπικών του δεδομένων. Στόχος ήταν τόσο η διευκόλυνση της καθημερινότητας του, όσο και το να αποκτήσει μια ακριβή εικόνα σχετικά με την ενέργεια που καταναλώνει, στο χαμηλότερο δυνατό κόστος κατασκευής.

Στο σύστημα υπάρχουν δύο βασικά είδη χρηστών. Ο διαχειριστής, ο οποίος μπορεί να ελέγχει τους λογαριασμούς των χρηστών και τις εγγεγραμμένες συσκευές, και οι απλοί χρήστες, οι οποίοι μπορούν να συνδέσουν συσκευές με το λογαριασμό τους, να ελέγχουν τις συσκευές τους και να προβάλουν στατιστικά και διαγράμματα σχετικά με την ενεργειακή τους κατανάλωση. Εκτός αυτών, υπάρχει η δυνατότητα διαμοιρασμού πρόσβασης μιας συσκευής σε άλλους χρήστες μέσω email.

Ωστόσο, ως χρήστες του συστήματος θα μπορούσαν να θεωρηθούν και οι ίδιες οι συσκευές. Ο ρόλος τους είναι, η επικοινωνία με το χρήστη μέσω της εφαρμογής, για την σύνδεση στο τοπικό δίκτυο, η αποστολή των δεδομένων κατανάλωσης που καταγράφουν στον εξυπηρετητή και η ανταπόκριση στις εντολές του χρήστη για τον απομακρυσμένο έλεγχο.

6.2 Μετρικά συστήματος

Το σύστημα αποτελείται τόσο από υλικό μέρος, όσο και από λογισμικό. Πρέπει λοιπόν, να αναφερθούν οι διάφορες παράμετροι που χαρακτηρίζουν τα δύο αυτά μέρη του συστήματος. Παρακάτω, παρουσιάζονται μετρήσεις σχετικές με τις δύο αυτές συνιστώσες.

6.2.1 Μετρικά υλικού μέρους

Στον παρακάτω πίνακα (Πίνακας 7) παρουσιάζονται οι μετρήσεις που σχετίζονται με το υλικό μέρος των συσκευών.

Υποσύστημα	Βάρος	Διαστάσεις
Μικροελεγκτής ESP-32	10 gr	5,5cm x 2,8cm
Μετρητής Κατανάλωσης SDM120M	90 gr	6,2cm x 11,9cm x 1,7cm
Ρελέ 5 volt	15 gr	4cm x 2,5cm
Μετατροπέας TTL σε RS485	5 gr	4,4cm x 1,4cm
Κουτί Κατασκευής	550 gr	22,5cm x 12cm x 10cm

Πίνακας 7 : Πίνακας μετρικών υλικού συστήματος

6.2.2 Μετρικά λογισμικού μέρους

Στον παρακάτω πίνακα (Πίνακας 8) παρουσιάζονται, οι μετρήσεις των αρχείων και των γραμμών του πηγαίου κώδικα για το λογισμικό του συστήματος, στα τμήματα της συσκευής, του εξυπηρετητή, της κινητής εφαρμογής και της εφαρμογής του διαχειριστή.

Υποσύστημα	Αριθμός αρχείων	Αριθμός γραμμών
Λογισμικό συσκευής	1	379
Λογισμικό εξυπηρετητή	21	1432
Λογισμικό κινητής εφαρμογής	36	6118
Λογισμικό εφαρμογής διαχειριστή	20	621

Πίνακας 8 : Πίνακας μετρικών λογισμικού μέρους

Για τις μετρήσεις χρησιμοποιήθηκε το εργαλείο CLOC[11] και δεν συμπεριλήφθηκαν τα σχόλια, οι κενές γραμμές, τα αρχεία ρυθμίσεων (configuration files), τα αρχεία καταγραφών (Log files) και οι εξωτερικές βιβλιοθήκες.

6.2.3 Το κόστος του συστήματος

Στην υποενότητα αυτή, γίνεται αναφορά στα επιμέρους κόστη και στο συνολικό ποσό που δαπανήθηκε για την υλοποίηση του συστήματος. Ο Πίνακας 9, παρουσιάζει το κόστος του κάθε τμήματος της συσκευής.

Υλικό	Κόστος
Μικροελεγκτής ESP32	8,00 ευρώ
Relay 5v (ρελέ)	2,20 ευρώ
Μετρητής κατανάλωσης SDM120M	50,00 ευρώ
Μετατροπέας MAX485 TTL σε RS485	1,20 ευρώ
Ηλεκτρολογικό κουτί IDE	15,00 ευρώ
Πρίζα ράγας τύπου Schuko	3,00 ευρώ
Κουμπί επανεκκίνησης	0,90 ευρώ
Κουμπί εναλλαγής λειτουργίας	0,70 ευρώ
Διάτρητες πλακέτες	8,90 ευρώ

Πίνακας 9 : Πίνακας ανάλυσης κόστους της συσκευής

[11] https://codetabs.com/count-loc/count-loc-online.html

Το συνολικό κόστος των υλικών του πίνακα, είναι 89,90 ευρώ. Διευκρινίζεται πως η προμήθεια των υλικών έγινε από ελληνικά καταστήματα.

6.3 Ανάλυση SWOT

Σε αυτή την ενότητα, γίνεται η ανάλυση SWOT για το σύστημα που υλοποιήθηκε. Εξετάζονται δηλαδή, τα δυνατά του σημεία (Strengths), οι αδυναμίες του (Weaknesses), οι ευκαιρίες που παρουσιάζονται (Opportunities) αλλά και οι πιθανές απειλές (Threats). Η σύνοψη των παραπάνω σε επιγραμματικές λίστες, βοηθά στην εξαγωγή συμπερασμάτων, καθώς και στον καθορισμό των μελλοντικών λειτουργικών απαιτήσεων.

Δυνατά σημεία:

- Παρακολούθηση της κατανάλωσης μιας συσκευής σε πραγματικό χρόνο.
- Εύκολη πρόσβαση στις συσκευές και τα δεδομένα τους μέσω κινητής εφαρμογής.
- Εύκολη ενσωμάτωση οποιουδήποτε μετρητή κατανάλωσης που υποστηρίζει το πρωτόκολλο Modbus. Για παράδειγμα, θα μπορούσε να ενσωματωθεί κάποιος τριφασικός μετρητής για βιομηχανική χρήση.
- Υψηλή ακρίβεια μέτρησης της κατανάλωσης με απόκλιση της τάξης του 1%.
- Ανοιχτού κώδικα (open-source), ώστε ο καθένας να μπορεί να υλοποιήσει το σύστημα μόνος του αλλά και να το βελτιώσει.

Αδυναμίες:

- Για την χρήση της εφαρμογής απαιτείται σύνδεση στο διαδίκτυο.
- Η συσκευή που υλοποιήθηκε είναι αρκετά ογκώδης, σε σύγκριση με άλλα συστήματα.

Ευκαιρίες:

- Δημιουργία ενός γενικότερου πλαισίου εφαρμογής, για την ένταξη πολλών διαφορετικών συσκευών IoT στο σύστημα όπως είναι συσκευές για μέτρηση θερμοκρασίας και υγρασίας ή έξυπνοι λαμπτήρες.
- Επανασχεδιασμός της συσκευής, ώστε να γίνει λιγότερο ογκώδης.
- Αλλαγή της βάσης δεδομένων MongoDB με μια βάση χρονοσειρών (timeseries), οι οποίες ειδικεύονται στην συλλογή, την αποθήκευση, την ανάκτηση, αλλά και την οπτικοποίηση μεγάλου όγκου δεδομένων που φέρουν χρονική σήμανση (timestamp).

Απειλές:

- Ύπαρξη πλήθους εμπορικών συσκευών σε χαμηλή τιμή.
- Εκμετάλλευση πιθανών αδυναμιών στην ασφάλεια του συστήματος, για τον έλεγχο των συσκευών των χρηστών από τρίτους.

6.4 Εκτίμηση κλιμάκωσης συστήματος

Καθώς το σύστημα σχεδιάστηκε με σκοπό να υποστηρίξει πολλαπλούς χρήστες και πολλαπλές συσκευές, είναι απαραίτητο να γίνει μια εκτίμηση συνεχόμενης λειτουργίας του σε βάθος πέντε χρόνων.

Αρχικά, θα εκτιμηθεί ο αποθηκευτικός χώρος που απαιτείται σε περίπτωση που ένας χρήστης ο οποίος διαθέτει τρεις συσκευές, χρησιμοποιεί το σύστημα για πέντε συνεχόμενα χρόνια. Η αποθήκευση ενός χρήστη στην βάση δεδομένων απαιτεί 35kB, η αποθήκευση μιας συσκευής απαιτεί 75kB, ενώ ο όγκος των δεδομένων που παράγει μια συσκευή σε 24 ώρες είναι περίπου 20kB. Έτσι προκύπτει, πως σε ένα έτος ο όγκος των δεδομένων για το χρήστη θα είναι:

$$35kB + 75kB + 3\ συσκευές * 20kB * 365\ ημέρες = 22\ MB$$

Άρα σε πέντε έτη ο όγκος των δεδομένων υπολογίζεται ως:

$$35kB + 75kB + 3\ συσκευές * 20kB * 365\ ημέρες * 5\ έτη = 109.5\ MB$$

Αν οι χρήστες του συστήματος αυξηθούν σε εκατό και οι συσκευές σε τριακόσιες, με βάση τα παραπάνω, ο αποθηκευτικός χώρος που απαιτείται είναι περίπου 11 GB. Σε αυτά δεν έχουν συμπεριληφθεί τα δεδομένα καταγραφής συμβάντων (συλλογή logs). Πρακτικά, η συλλογή logs μπορεί περιοδικά να διαγράφεται.

Όσον αφορά την δυνατότητα διεκπεραίωσης των αιτημάτων HTTP και των μηνυμάτων MQTT, αυτή εξαρτάται σε μεγάλο βαθμό από τα χαρακτηριστικά του φυσικού μηχανήματος, στο οποίο εκτελείται το λογισμικό του εξυπηρετητή και η βάση δεδομένων MongoDB. Ωστόσο, κάποια στοιχειώδη στατιστικά δεδομένα σχετικά με το MQTT, αναφέρονται στην σελίδα της βιβλιοθήκης λογισμικού Aedes.js[12].

6.5 Μελλοντικές επεκτάσεις

Στο σύστημα που αναπτύχθηκε, υπάρχει δυνατότητα για αρκετές μελλοντικές επεκτάσεις. Οι επεκτάσεις αυτές, έχουν να κάνουν με την προσθήκη νέων λειτουργιών, οι οποίες μπορούν να ενταχθούν στο υπάρχων σύστημα με μερικές τροποποιήσεις.

Μια προσθήκη στις λειτουργίες του, θα μπορούσε να είναι ο χρονοπρογραμματισμός της ενεργοποίησης/απενεργοποίησης των συσκευών. Για παράδειγμα, ο χρήστης θα μπορούσε να θέσει την ώρα στην οποία επιθυμεί να ανάβουν και να σβήνουν τα φώτα. Οι εφαρμογές της λειτουργίας αυτής είναι ποικίλες και αλλάζουν ανάλογα τις ανάγκες του κατόχου της συσκευής.

Άλλη μια επέκταση, αφορά την προσθήκη αισθητήρων στις συσκευές. Για παράδειγμα, χρησιμοποιώντας αισθητήρες θερμοκρασίας ή υγρασίας ο χρήστης θα μπορούσε να παρακολουθεί την ενεργειακή απόδοση του χώρου του ή ακόμα και να ενεργοποιούνται οι κατάλληλες συσκευές αυτόματα σε περίπτωση υπέρβασης κάποιον ορίων. Παραδείγματος χάριν, οι υψηλές τιμές υγρασίας να ενεργοποιούν έναν αφυγραντήρα.

Μια εξαιρετική προσθήκη στο κομμάτι του λογισμικού, θα ήταν η χρήση ενός αλγορίθμου μηχανικής μάθησης. Ο αλγόριθμος αυτός σε βάθος χρόνου θα μπορούσε να

[12] https://github.com/moscajs/aedes#acknowledgements

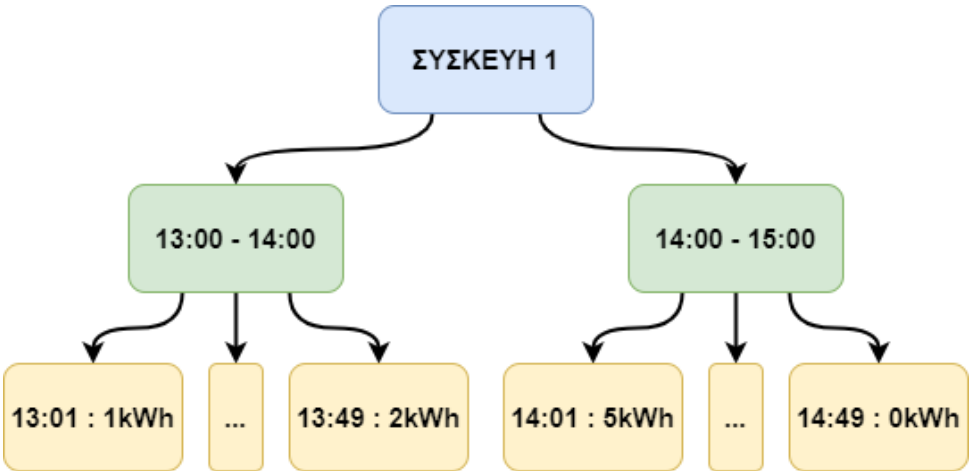
προβλέπει την πιθανή ενεργειακή κατανάλωση του χρήστη για κάθε μήνα ή ακόμα και να του προτείνει αλλαγές, οι οποίες μπορούν να του εξοικονομήσουν χρήματα, βασισμένες σε δεδομένα τα οποία μπορούν να συλλεχθούν από τους υπόλοιπους χρήστες του συστήματος.

Τέλος, στο θέμα της γρηγορότερης αποθήκευσης και ανάκτησης των δεδομένων που παράγουν οι συσκευές, θα μπορούσαν να χρησιμοποιηθούν στην βάση δεδομένων MondoDB, οι τεχνικές Sharding και Replica Sets που αναφέρθηκαν στο δεύτερο κεφάλαιο.

6.6 Προβλήματα που αντιμετωπίστηκαν

Κατά την ανάπτυξη του συστήματος, προέκυψαν αρκετά προβλήματα, κάποια από τα οποία αντιμετωπίστηκαν, ενώ κάποια άλλα δεν υπήρξε η δυνατότητα και ο χρόνος να επιλυθούν στα πλαίσια αυτής της διπλωματικής.

Ένα από τα βασικά προβλήματα που αντιμετωπίστηκαν, είναι η διαχείριση τόσο μεγάλου όγκου δεδομένων. Συγκεκριμένα, ο αριθμός των καταγραφών έγινε πολύ γρήγορα πολύ μεγάλος, γεγονός το οποίο προκαλούσε καθυστερήσεις και προβλήματα απόδοσης στην ανάκτηση τους από την βάση δεδομένων και στην παρουσίαση τους στο χρήστη. Η λύση σε αυτό ήταν η δημιουργία διαφορετικών συλλογών κατανάλωσης στην βάση δεδομένων για κάθε συσκευή. Επιπλέον, χρησιμοποιήθηκε μια τεχνική ομαδοποίησης των καταγραφών (data bucketing), στην οποία τα δεδομένα οργανώνονται σε ομάδες της μίας ώρας. Έτσι, κατά την αναζήτηση μιας μέτρησης αρκεί να βρεθεί το χρονικό «παράθυρο» στο οποίο αυτή βρίσκεται. Με τον τρόπο αυτό, ο χρόνος αναζήτησης της κατανάλωσης για ένα χρονικό εύρος μειώνεται σημαντικά. Ένα διάγραμμα για την κατανόηση της τεχνικής αυτής παρουσιάζεται στην Εικόνα 80.



Εικόνα 80 : Οπτικοποίηση της τεχνικής αποθήκευσης δεδομένων Bucketing

Άλλη μια δυσκολία που προέκυψε ήταν η έλλειψη ποικιλίας και υποστήριξης στις διαθέσιμες βιβλιοθήκες κώδικα για το Bluetooth χαμηλής ενέργειας (BLE). Συγκεκριμένα, έγιναν πολλές δοκιμές με διαφορετικές βιβλιοθήκες κώδικα μέχρι να επιτευχθεί επιτυχής επικοινωνία ανάμεσα στην εφαρμογή και τη συσκευή, κάτι το οποίο κόστισε σε χρόνο υλοποίησης.

Τέλος, το αρχικό πλάνο ήταν να χρησιμοποιηθεί για την μέτρηση της κατανάλωσης ο αισθητήρας ACS712 του οποίου το κόστος είναι εξαιρετικά χαμηλό. Ωστόσο, μέσω

πειραματικών διαδικασιών διαπιστώθηκε πως ο αισθητήρας αυτός ήταν αναξιόπιστος και ανακριβής για την χρήση του σε αυτή την εργασία. Έτσι, επιλέχθηκε η χρήση ενός εμπορικού μετρητή που υποστηρίζει το αξιόπιστο βιομηχανικό πρωτόκολλο Modbus.

6.7 Τα στάδια της έρευνας

Τα διακριτά στάδια της έρευνας για την διαμόρφωση των λειτουργιών, την εύρεση των απαραίτητων τεχνολογιών, αλλά και των τεχνικών διαχείρισης των δεδομένων για την μέγιστη αποδοτικότητα του συστήματος είναι τα εξής:

- Αναζήτηση πλατφόρμας υλικού με υποστήριξη σύνδεσης σε ασύρματο δίκτυο (WiFi).
- Αναζήτηση τεχνολογιών που επιτρέπουν την δημιουργία εφαρμογών για διαφορετικές πλατφόρμες με κοινή βάση κώδικα (cross platform) για την ανάπτυξη της κινητής εφαρμογής και της εφαρμογής διαχειριστή.
- Αναζήτηση έμπνευσης για την εμφάνιση των διεπαφών των χρηστών και σχεδιασμός τους με βάση τις λειτουργικές απαιτήσεις.
- Εύρεση της κατάλληλης βάσης δεδομένων, με βάση τον όγκο των δεδομένων που παράγονται από τις συσκευές.
- Έρευνα για την εύρεση της κατάλληλης Back-End τεχνολογίας, για την γρήγορη διεκπεραίωση μεγάλου όγκου αιτημάτων.
- Αναζήτηση τεχνικών αποθήκευσης των δεδομένων στην βάση, με στόχο την γρήγορη ανάκτηση τους, για την δημιουργία στατιστικών και γραφημάτων.
- Αναζήτηση βιβλιοθηκών ανοιχτού κώδικα οι οποίες να υλοποιούν το πρωτόκολλο Modbus, για την επίτευξη επικοινωνίας με το μετρητή κατανάλωσης.
- Έρευνα για την εξασφάλιση της ακεραιότητας του χρήστη, κατά την χρήση της συσκευής.
- Έρευνα για την εφαρμογή τεχνικών ασφαλείας από κακόβουλους χρήστες.

6.8 Συμπεράσματα

Το σύστημα που υλοποιήθηκε είναι πολυδιάστατο, με αποτέλεσμα η ομαλή λειτουργία του να εξαρτάται από διαφορετικά υποσυστήματα. Παρόλο που με την υπάρχουσα μορφή του είναι αξιόπιστο και καλύπτει πολλαπλά σενάρια χρήσης, είναι σίγουρο ότι σε πολλές πτυχές του μπορεί να δεχθεί σημαντικές αναβαθμίσεις. Ωστόσο, με βάση τα όσα αναλύθηκαν στην διπλωματική αυτή, το σύστημα επιτυγχάνει το σκοπό του. Βοηθά το χρήστη τόσο στην καθημερινότητα του, όσο και σε βάθος χρόνου, επιτρέποντας του να κατανοήσει την ενεργειακή απόδοση των συσκευών του και να λάβει δράση για να την βελτιώσει. Τέλος, ο οποιοσδήποτε έχει τη δυνατότητα να υλοποιήσει και να χρησιμοποιήσει το παρών σύστημα, χωρίς να εξαρτάται από τρίτους για την λειτουργία του.

Παράρτημα – Εγκατάσταση Λογισμικού

Σε αυτό το τμήμα, αναφέρεται η διαδικασία για την εγκατάσταση των λογισμικών που αναπτύχθηκαν στα πλαίσια αυτής της διπλωματικής εργασίας, σε τοπικό περιβάλλον. Προαπαιτούμενα είναι η εγκατάσταση των εξής πακέτων σε υπολογιστή με λειτουργικό σύστημα Windows ή Linux (το λογισμικό αυτής της διπλωματικής εργασίας, αναπτύχθηκε σε περιβάλλον windows):

- Εγκατάσταση του περιβάλλοντος Node.js[13]
- Εγκατάσταση της βάσης δεδομένων MongoDB[14]
- Εγκατάσταση του συστήματος διαχείρισης εκδόσεων git[15]
- Εγκατάσταση των απαραίτητων πακέτων με βάση τον οδηγό[16] στην ιστοσελίδα του React Native
- Από ένα τερματικό (π.χ. cmd, bash, κλπ.), εκτέλεση της εντολής *git clone https://github.com/panbak/HomeIO.git*

Οδηγίες εγκατάστασης λογισμικού διακομιστή

Η διαδικασία εγκατάστασης του λογισμικού του διακομιστή ακολουθεί τα εξής βήματα:

- Εκτέλεση της εντολής *cd HomeIO_Server* , για να ανοίξει ο φάκελος με τον κώδικα του διακομιστή
- Εκτέλεση της εντολής *npm install*, για να εγκατασταθούν τα απαραίτητα πακέτα
- Επεξεργασία του αρχείου *.env.example,* για ρύθμιση των μεταβλητών περιβάλλοντος και μετονομασία του σε *.env*
- Εκτέλεση της εντολής *npm start,* για να τρέξει το λογισμικό του διακομιστή

[13] https://nodejs.org/en/download/
[14] https://www.mongodb.com/try/download/community?tck=docs_server
[15] https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
[16] https://reactnative.dev/docs/environment-setup

Οδηγίες εγκατάστασης διαδικτυακής εφαρμογής διαχειριστή

Η διαδικασία εγκατάστασης της εφαρμογής του διαχειριστή ακολουθεί τα εξής βήματα:

- Εκτέλεση της εντολής *cd HomeIO_Admin*, για να ανοίξει ο φάκελος με τον κώδικα της εφαρμογής διαχειριστή
- Εκτέλεση της εντολής *npm install*, για να εγκατασταθούν τα απαραίτητα πακέτα
- Επεξεργασία του αρχείου *src/config.js,* για ρύθμιση του url του διακομιστή
- Εκτέλεση της εντολής *npm start,* για να τρέξει το λογισμικό του διακομιστή

Για την εγκατάσταση των πιστοποιητικών SSL, προτείνετε η χρήση του εργαλείου certbot[17].

Οδηγίες εγκατάστασης κινητής εφαρμογής

Η διαδικασία εγκατάστασης της κινητής εφαρμογής ακολουθεί τα εξής βήματα:

- Εκτέλεση της εντολής *cd HomeIO_App*, για να ανοίξει ο φάκελος με τον κώδικα της κινητής εφαρμογής
- Εκτέλεση της εντολής *npm install*, για να εγκατασταθούν τα απαραίτητα πακέτα
- Επεξεργασία του αρχείου περιβάλλοντος *environments.js,* για ρύθμιση των url ανάπτυξης (development) και παραγωγής (production).
- Αν χρησιμοποιηθεί φυσική συσκευή και όχι εξομοιωτής(emulator), πρέπει να γίνουν πρώτα τα βήματα του οδηγού[18] από την ιστοσελίδα του React Native
- Εκτελούμε την εντολή *npx react-native run-android*

Για την δημιουργία εκτελέσιμου αρχείου apk η διαδικασία είναι:

- Εκτέλεση της εντολής *keytool -genkey -v -keystore your_key_name.keystore -alias your_key_alias -keyalg RSA -keysize 2048 -validity 10000* για την δημιουργία αρχείου κλειδιού.
- Μεταφορά του αρχείου κλειδιού μέσα στο φάκελο */android/app.*
- Εκτέλεση της εντολής *react-native bundle --platform android --dev false --entry-file index.js --bundle-output android/app/src/main/assets/index.android.bundle --assets-dest android/app/src/main/res/ .*
- Εκτέλεση της εντολής *cd android*
- Εκτέλεση της εντολής *.\gradlew assembleRelease*
- Το αρχείο apk της εφαρμογής, θα βρίσκεται στην τοποθεσία *android/app/build/outputs/apk/release/app-release.apk.*

Compile και ανέβασμα κώδικα στον μικροεπεξεργαστή ESP-32

Για την συγγραφή κώδικα στο Arduino IDE για το ESP-32, πρέπει να γίνει η εξής διαδικασία:

- Αφού ανοίξει το Arduino IDE, στην επιλογή *menu > File > Preferences*
- Στο πεδίο "*Additional Board Manager URLs*" πρέπει να γίνει επικόλληση του : *https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json*. Επιλέξτε 'OK'.
- Στην συνέχεια, στο *menu > Tools > Board > Board Managers*, να γίνει αναζήτηση του '*esp32*' και να επιλεχθεί η επιλογή '*install*' για το '*ESP32 by Espressif Systems*'.
- Τέλος, στο menu > Tools > Board, επιλέξτε την πλατφόρμα ESP-32 που επιθυμείτε.

[17] https://certbot.eff.org/
[18] https://reactnative.dev/docs/running-on-device

Για την εγκατάσταση της βιβλιοθήκης SDM:

- Λήψη του πηγαίου κώδικα της βιβλιοθήκης (έκδοση 2.2.0) από το Github[19].
- Στο Arduino IDE στο *menu > Sketch > Include Library > Add .zip Library* και επιλογή του αρχείου zip που λήφθηκε στο προηγούμενο βήμα.
- Εύρεση και άνοιγμα του Sketchbook Location (*File > Preferences > Sketchbook*) στον τοπικό δίσκο.
- Κατεύθυνση στο *Libraries > SDM_ENERGY_METER* και επεξεργασία του αρχείου *SDM_Config_User.h*
- Προσθήκη των γραμμών:
  - *#define USE_HARDWARESERIAL*
  - *#define SDM_UART_BAUD 2400*
  - *#define ESP32*

- Αλλαγή του τμήματος:

```
#if defined ( USE_HARDWARESERIAL )
  #if defined ( ESP32 )
    #define SDM_RX_PIN          13
    #define SDM_TX_PIN          15
  #endif
#else
```

σε:

```
#if defined ( USE_HARDWARESERIAL )
  #if defined ( ESP32 )
    #define SDM_RX_PIN          16
    #define SDM_TX_PIN          17
  #endif
#else
```

Επίλυση προβλημάτων

Αν κατά την εκτέλεση της εντολής *npm start,* για το λογισμικό του διακομιστή ή της εφαρμογής του διαχειριστή, προκύψει κάποιο σφάλμα, μπορεί να εκτελεστεί η εντολή *npm audit fix,* για προσπάθεια αυτόματης επίλυσης.

Αν προκύψει οποιοδήποτε πρόβλημα κατά την εγκατάσταση ή την εκτέλεση της κινητής εφαρμογής, να ελεγχθούν τα βήματα της επίλυσης προβλημάτων[20] από την ιστοσελίδα του React Native.

[19] https://github.com/reaper7/SDM_Energy_Meter/releases/tag/v2.2.0
[20] https://reactnative.dev/docs/troubleshooting

Βιβλιογραφία

[1] M. R. Y. Jamil Y. Khan, «Internet of Things (IoT): Systems and Applications,» CRC Press, 2019.

[2] «Tapo P110,» [Ηλεκτρονικό]. Available: https://www.tp-link.com/gr/home-networking/smart-plug/tapo-p110/. [Πρόσβαση Σεπτέμβριος 2022].

[3] «Amazon Smart Plug,» [Ηλεκτρονικό]. Available: https://www.amazon.com/Amazon-smart-plug-works-with-Alexa/dp/B089DR29T6. [Πρόσβαση Σεπτέμβριος 2022].

[4] «Shelly Plug,» [Ηλεκτρονικό]. Available: https://shelly.cloud/products/shelly-plug-smart-home-automation-device/. [Πρόσβαση Σεπτέμβριος 2022].

[5] «Evolution Of HTTP,» [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP. [Πρόσβαση Άυγουστος 2022].

[6] «HTTPS,» [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Glossary/https. [Πρόσβαση Σεπτέμβριος 2022].

[7] OASIS, «MQTT,» [Ηλεκτρονικό]. Available: https://mqtt.org/. [Πρόσβαση Άυγουστος 2022].

[8] Bluetooth, «Intro to Bluetooth GATT,» [Ηλεκτρονικό]. Available: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-gap-gatt/. [Πρόσβαση Άυγουστος 2022].

[9] «Modbus FAQ,» [Ηλεκτρονικό]. Available: https://www.modbus.org/faq.php. [Πρόσβαση Σεπτέμβριος 2022].

[10] W3Schools, «JavaScript History,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/js/js_history.asp. [Πρόσβαση Άυγουστος 2022].

[11] nodejs.org, «About Node.js,» [Ηλεκτρονικό]. Available: https://nodejs.org/en/about/. [Πρόσβαση Άυγουστος 2022].

[12] «Express.js,» [Ηλεκτρονικό]. Available: https://expressjs.com/. [Πρόσβαση Άυγουστος 2022].

[13] «Mongoose.js,» [Ηλεκτρονικό]. Available: https://mongoosejs.com/. [Πρόσβαση Άυγουστος 2022].

[14] «AEDES,» 26 07 2022. [Ηλεκτρονικό]. Available: https://github.com/moscajs/aedes#readme. [Πρόσβαση Άυγουστος 2022].

[15] «JSON Web Token Introduction,» [Ηλεκτρονικό]. Available: https://jwt.io/introduction. [Πρόσβαση Σεπτέμβριος 2022].

[16] N. P. &. D. Mazieres, «A Future-Adaptable Password Scheme,» 28 04 1999. [Ηλεκτρονικό]. Available: https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node1.html. [Πρόσβαση Άυγουστος 2022].

[17] «React Native: Bringing modern web techniques to mobile,» [Ηλεκτρονικό]. Available: https://reactnative.dev/blog/2015/03/26/react-native-bringing-modern-web-techniques-to-mobile. [Πρόσβαση Σεπτέμβριος 2022].

[18] «React Navigation,» [Ηλεκτρονικό]. Available: https://reactnavigation.org/. [Πρόσβαση Άυγουστος 2022].

[19] «Async Storage,» [Ηλεκτρονικό]. Available: https://react-native-async-storage.github.io/async-storage/. [Πρόσβαση Άυγουστος 2022].

[20] «React Native MQTT,» [Ηλεκτρονικό]. Available: https://www.npmjs.com/package/@taoqf/react-native-mqtt. [Πρόσβαση Σεπτέμβριος 2022].

[21] «React Native Ble plx,» [Ηλεκτρονικό]. Available: https://dotintent.github.io/react-native-ble-plx/. [Πρόσβαση Άυγουστος 2022].

[22] Ν. Μ. Χατζηγιαννάκης, σε *Η γλώσσα C σε βάθος - 4η αναθεωρημένη έκδοση*, Κλειδάριθμος.

[23] «Η προέλευση της γλώσσας C++,» σε *C++ Επίλυση Προβλημάτων - 9η έκδοση*, Εκδόσεις Τζιόλα.

[24] «SDM Energy Meter,» [Ηλεκτρονικό]. Available: https://github.com/reaper7/SDM_Energy_Meter/. [Πρόσβαση Σεπτέμβριος 2022].

[25] «How To Scale MongoDB,» [Ηλεκτρονικό]. Available: https://www.mongodb.com/basics/scaling. [Πρόσβαση Σεπτέμβριος 2022].

[26] «Visual Studio Code - Code Edititng Redefined,» [Ηλεκτρονικό]. Available: https://code.visualstudio.com/. [Πρόσβαση Σεπτέμβριος 2022].

[27] «Software | Arduino,» [Ηλεκτρονικό]. Available: https://www.arduino.cc/en/software. [Πρόσβαση Σεπτέμβριος 2022].

[28] «Android Studio,» [Ηλεκτρονικό]. Available: https://developer.android.com/studio. [Πρόσβαση Άυγουστος 2022].

[29] ESPRESSIF, «ESP32,» [Ηλεκτρονικό]. Available: https://www.espressif.com/en/products/socs/esp32. [Πρόσβαση Άυγουστος 2022].

[30] «SDM120M Series,» [Ηλεκτρονικό]. Available: https://www.eastroneurope.com/products/view/sdm120modbus. [Πρόσβαση Σεπτέμβριος 2022].

[31] «MDN Web Docs,» 31 Ιούλιος 2022. [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Glossary/https. [Πρόσβαση 05 Αύγουστος 2022].

[32] «Code Editing. Redefined.,» [Ηλεκτρονικό]. Available: https://github.com/mqttjs/MQTT.js#readme. [Πρόσβαση Άυγουστος 2022].

[33] «Introduction to JWT,» [Ηλεκτρονικό]. Available: https://jwt.io/introduction. [Πρόσβαση Άυγουστος 2022].

[34] «MDN Web Docs,» 15 Ιουλίου 2022. [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Πρόσβαση Αύγουστος 2022].

[35] «MQTT,» [Ηλεκτρονικό]. Available: https://github.com/mqttjs/MQTT.js#readme. [Πρόσβαση Άυγουστος 2022].

[36] «The open-source Arduino Software,» [Ηλεκτρονικό]. Available: https://www.arduino.cc/en/software. [Πρόσβαση Άυγουστος 2022].