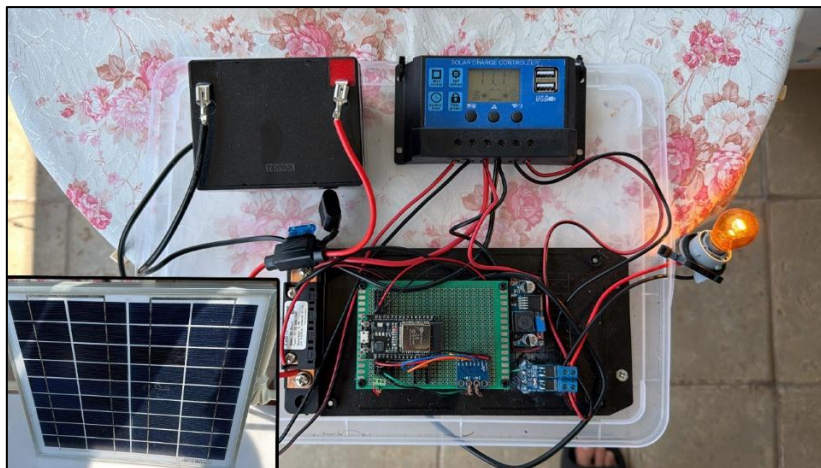


ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και Ανάπτυξη Συστήματος IoT σε Python
για την Παρακολούθηση της Υγείας SoH και της
φόρτισης (SoC) Μπαταριών Φωτοβολταϊκών σε
Πραγματικό Χρόνο»



Του φοιτητή
Πλαγιαρινού Δημήτριου
Αρ. Μητρώου: 2021135

Επιβλέπων
Κοκκώνης Γεώργιος
Αναπληρωτής Καθηγητής

Ημερομηνία ΣΕΠΤΕΜΒΡΙΟΣ 2026

Τίτλος Δ.Ε. «Σχεδιασμός και Ανάπτυξη Συστήματος IoT σε Python για την Παρακολούθηση της Υγείας SoH και της φόρτισης (SoC) Μπαταριών Φωτοβολταϊκών σε Πραγματικό Χρόνο»

Κωδικός Δ.Ε. 26132

Ονοματεπώνυμο φοιτητή: Πλαγιαρινός Δημήτριος

Ονοματεπώνυμο εισηγητή: Κοκκώνης Γεώργιος

Ημερομηνία ανάληψης Δ.Ε. 02/03/2026

Ημερομηνία περάτωσης Δ.Ε. ΣΕΠΤΕΜΒΡΙΟΣ 2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Πλαγιαρινού Δημήτρη που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στην οικογένεια μου και σε όλους μου τους φίλους.»

Πρόλογος

Ονομάζομαι Πλαγιαρινός Δημήτρης και η εγγραφή μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε. έγινε τον Οκτώβριο του 2021. Φτάνοντας στο τέλος των σπουδών μου, επέλεξα το συγκεκριμένο θέμα διπλωματικής εργασίας καθώς συνδυάζει αρμονικά τα δύο βασικά επιστημονικά πεδία του τμήματος: την επιστήμη των Ηλεκτρονικών και την Πληροφορική. Η εκπόνηση της παρούσας εργασίας αποτέλεσε για εμένα την ευκαιρία να συνθέσω τις θεωρητικές και τεχνικές γνώσεις που απέκτησα κατά τη διάρκεια της φοίτησής μου, μετουσιώνοντάς τες σε μια ολοκληρωμένη και λειτουργική λύση. Μέσα από τη διασύνδεση των ενσωματωμένων συστημάτων, των πρωτοκόλλων δικτύωσης και των διαδικτυακών πλατφορμών, στόχος μου ήταν να αναπτύξω ένα σύστημα χαμηλού κόστους που απαντά σε πραγματικές ανάγκες της σύγχρονης ενεργειακής διαχείρισης.

Περίληψη

Η παρούσα εργασία πραγματεύεται τον σχεδιασμό, την ανάπτυξη και την υλοποίηση ενός ολοκληρωμένου συστήματος Διαδικτύου των Πραγμάτων (IoT) για την παρακολούθηση σε πραγματικό χρόνο της Κατάστασης Φόρτισης (SoC) και της Κατάστασης Υγείας (SoH) μπαταριών σε αυτόνομα φωτοβολταϊκά συστήματα. Σκοπός της κατασκευής είναι η δημιουργία μιας αξιόπιστης και κλιμακώσιμης λύσης που βελτιστοποιεί τη διαχείριση της αποθηκευμένης ενέργειας, προστατεύει τους συσσωρευτές, συμβάλλοντας στη δυναμική παράταση της διάρκειας ζωής τους. Η αρχιτεκτονική του συστήματος ακολουθεί πιστά το μοντέλο των τριών επιπέδων. Στο Επίπεδο Αντίληψης (Hardware), ο πυρήνας του συστήματος είναι ο μικροελεγκτής ESP32, ο οποίος συλλέγει τηλεμετρικά δεδομένα υψηλής ακρίβειας. Η μέτρηση της τάσης και του ρεύματος επιτυγχάνεται μέσω του αισθητήρα INA226 και μιας εξωτερικής αντίστασης διακλάδωσης (Shunt 50A), ενώ η θερμοκρασία καταγράφεται από τον αισθητήρα DS18B20. Παράλληλα, ο μικροελεγκτής λειτουργεί ως Τοπικό Σύστημα Διαχείρισης (Edge BMS), ελέγχοντας δυναμικά την εκφόρτιση μέσω διάταξης διπλού MOSFET για την προστασία της μπαταρίας από ακραίες συνθήκες λειτουργίας. Στο Επίπεδο Δικτύου, τα δεδομένα μορφοποιούνται σε πακέτα JSON και μεταδίδονται ασύρματα μέσω του πρωτοκόλλου HTTP POST. Έχει δοθεί ιδιαίτερη έμφαση στην ευρωστία του κώδικα (MicroPython), ενσωματώνοντας μηχανισμούς ανοχής σε σφάλματα (fault tolerance) και μηχανισμούς διαχείρισης μνήμης και ρητή συλλογή αχρησιμοποίητης μνήμης. Τέλος, στο Επίπεδο Εφαρμογής, η διαδικτυακή πλατφόρμα αναπτύχθηκε με το πλαίσιο Django (Python) και εκτελείται σε τοπικό περιβάλλον, αξιοποιώντας τη διαδικτυακή serverless βάση δεδομένων PostgreSQL της Neon. Οι αλγόριθμοι στο backend αναλύουν τα δεδομένα για τον υπολογισμό των δεικτών SoC και SoH. Τα αποτελέσματα οπτικοποιούνται σε μια σύγχρονη και δυναμική Διεπαφή Χρήστη με διαδραστικά γραφήματα (Chart.js), η οποία περιλαμβάνει εξατομικευμένο πίνακα ελέγχου πελάτη, κέντρο διαχείρισης στόλου (Fleet Management) και αυτοματοποιημένο σύστημα ειδοποιήσεων, προσφέροντας πλήρη απομακρυσμένη εποπτεία.

«Design and Development of an IoT System in Python for Monitoring the Health (SoH) and Charging (SoC) of Photovoltaic Batteries in Real Time»

«Dimitrios Plagiarinos»

Abstract

This thesis focuses on the design, development, and implementation of a comprehensive Internet of Things (IoT) system for the real-time monitoring of the State of Charge (SoC) and State of Health (SoH) of lead-acid batteries used in standalone photovoltaic (PV) systems. The primary objective is to develop a reliable and scalable solution that optimizes stored energy management while protecting battery banks, thereby extending their operational lifespan. The system architecture strictly follows the three-layer IoT model. At the Perception Layer (Hardware), the core of the system is the ESP32 microcontroller, which acquires high-precision telemetry data. Voltage and current measurements are obtained using an INA226 sensor in conjunction with an external 50 A shunt resistor, while battery temperature is monitored by a DS18B20 temperature sensor. In addition, the microcontroller functions as a Local Battery Management System (Edge BMS), dynamically controlling battery discharge through a dual-MOSFET switching circuit to protect the battery against extreme operating conditions. At the Network Layer, the collected data are formatted as JSON packets and transmitted wirelessly using the HTTP POST protocol. Particular emphasis has been placed on the robustness of the MicroPython firmware through the implementation of fault-tolerance mechanisms and dynamic memory management techniques. Finally, at the Application Layer, the web platform was developed using the Django framework (Python) and runs locally, leveraging Neon's serverless PostgreSQL database. Backend algorithms process the incoming telemetry data to estimate the battery's State of Charge (SoC) and State of Health (SoH). The computed results are presented through a modern, dynamic user interface featuring interactive Chart.js visualizations. The platform includes a personalized customer dashboard, a fleet management center, and an automated alert system, providing comprehensive remote monitoring and management capabilities.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον καθηγητή Γεώργιο Κοκκώνη για την καθοδήγηση, τη στήριξη και τις πολύτιμες παρατηρήσεις καθ' όλη τη διάρκεια της εκπόνησης της εργασίας μου.

Περιεχόμενα

Πρόλογος.....	v
Περίληψη.....	vi
Abstract	vii
Ευχαριστίες	viii
Περιεχόμενα	ix
Κατάλογος Σχημάτων	xiii
Κατάλογος Πινάκων.....	xv
Συντομογραφίες.....	xvi
Κεφάλαιο 1ο: Εισαγωγή	1
1.1 Πλαίσιο και κίνητρο.....	1
1.2 Ορισμός του προβλήματος.....	1
1.3 Διαμόρφωση κειμένου.....	1
1.4 Στόχοι και ερευνητικά ερωτήματα	2
1.5 Συνεισφορά.....	2
1.6 Δομή της εργασίας	2
1.7 Επίλογος.....	3
Κεφάλαιο 2ο: Τεχνολογίες του IoT: Ενσωματωμένα Συστήματα και Web APIs	4
2.1 Εισαγωγή.....	4
2.2 Διαδίκτυο των Πραγμάτων (IoT)	4
2.2.1 Αρχιτεκτονική του IoT	4
2.2.2 Χαρακτηριστικά του IoT	5
2.3 Ενσωματωμένα συστήματα.....	6
2.3.1 Επισκόπηση ES	6
2.3.2 Χαρακτηριστικά ES.....	7
2.3.3 Εφαρμογές Μικροελεγκτών σε Συστήματα IoT	8
2.4 WEB APIs	8
2.4.1 Αρχιτεκτονική και Λειτουργία Web APIs.....	8
2.4.2 Πρωτόκολλα και Μορφές Δεδομένων.....	9
2.4.3 Ο Ρόλος των Web APIs στα Συστήματα IoT	9
Κεφάλαιο 3ο: Θεωρητικό Υπόβαθρο Συσσωρευτών.....	10
3.1 Τεχνολογία Μπαταριών Μολύβδου-Οξέος (VRLA)	10
3.2 Στάδια Φόρτισης Φωτοβολταϊκών Συστημάτων.....	10

3.3	Μεθοδολογίες Εκτίμησης Κατάστασης Φόρτισης (SoC).....	11
3.4	Παράγοντες Υποβάθμισης (SoH).....	12
3.5	Βιβλιογραφική Ανασκόπηση Μεθόδων Εκτίμησης SoC και SoH.....	12
Κεφάλαιο 4ο:	Συνολική Αρχιτεκτονική του Συστήματος.....	14
4.1	Εισαγωγή.....	14
4.2	Χαρακτηριστικά του συστήματος	14
4.3	Γενική περιγραφή της εφαρμογής.....	15
4.4	Λειτουργικές και Μη Λειτουργικές Απαιτήσεις του Συστήματος	15
4.4.1	Λειτουργικές Απαιτήσεις.....	15
4.4.2	Μη Λειτουργικές Απαιτήσεις.....	16
4.5	Εναλλακτικές Αρχιτεκτονικές Επιλογές και Τεκμηρίωση Σχεδιαστικών Αποφάσεων	17
4.5.1	Επιλογή Μικροελεγκτή	18
4.5.2	Επιλογή Πρωτοκόλλου Επικοινωνίας	18
4.5.3	Επιλογή Υποδομής Αποθήκευσης Δεδομένων.....	18
4.5.4	Κατανομή Λειτουργικότητας μεταξύ Edge και Cloud	18
4.5.5	Σύγκριση με Συναφή Συστήματα IoT-BMS της Βιβλιογραφίας.....	19
Κεφάλαιο 5ο:	Εργαλεία Ανάπτυξης και Γλώσσες Προγραμματισμού	21
5.1	Εισαγωγή.....	21
5.2	MicroPython.....	21
5.3	Python.....	21
5.4	Τεχνολογίες Ιστού (HTML, JavaScript, Chart.js).....	23
5.5	PostgreSQL	24
5.6	Django Framework.....	24
5.7	Thonny IDE.....	24
5.8	Visual Studio Code.....	24
5.9	GitHub	25
5.10	Υποδομές Εκτέλεσης και Φιλοξενίας.....	25
Κεφάλαιο 6ο:	Επίπεδο Αντίληψης (Perception Layer).....	26
6.1	Εισαγωγή.....	26
6.2	Ο Μικροελεγκτής ESP32 NodeMCU-32S.....	26
6.3	Οι Αισθητήρες του Συστήματος.....	27
6.3.1	Ο αισθητήρας DS18B20.....	27
6.3.2	Ο αισθητήρας INA226 (CJMCU-226)	28
6.3.3	Το πρωτόκολλο I2C.....	29
6.4	Το Κύκλωμα Ισχύος.....	30

6.4.1	Φωτοβολταϊκό Πάνελ 12V (10W).....	30
6.4.2	Μπαταρία Μολύβδου 12V DC.....	31
6.4.3	Ελεγκτής Φόρτισης 30A PWM.....	31
6.4.4	Το Φορτίο (P21W-BA15S-1156).....	31
6.4.5	Μετατροπέας Τάσης (Step-Down Buck Converter).....	32
6.4.6	Μονάδα Ελέγχου Φορτίου (Dual High-Power MOSFET).....	33
6.5	Σχεδιασμός και Συνδεσμολογία Κυκλώματος.....	33
6.5.1	Το fritzing.....	33
6.5.2	Το Ηλεκτρονικό Κύκλωμα.....	34
6.5.3	Η Κατασκευή.....	36
Κεφάλαιο 7ο:	Επίπεδο Δικτύου (Network Layer).....	41
7.1	Εισαγωγή.....	41
7.2	Σύνδεση στο Δίκτυο Wi-Fi.....	41
7.3	Πρωτόκολλο Μεταφοράς Δεδομένων (HTTP POST).....	41
7.3.1	HTTP POST έναντι MQTT.....	42
7.4	Διαδικασία Αποστολής και Συγχρονισμού.....	43
7.5	Διάγραμμα ροής (Flow Chart) Προγράμματος ESP32.....	44
7.6	Εξήγηση Διαγράμματος Ροής.....	46
7.6.1	Αρχικοποίηση και Έλεγχος Δικτύου.....	46
7.6.2	Σάρωση Αισθητήρων και Εκκίνηση.....	46
7.6.3	Κύριος Βρόχος και Τοπικός Έλεγχος (Edge BMS).....	47
7.6.4	Αποστολή Δεδομένων και Διαχείριση Μνήμης.....	47
Κεφάλαιο 8ο:	Επίπεδο Εφαρμογής (Application Layer).....	48
8.1	Εισαγωγή.....	48
8.2	Αρχιτεκτονική του Django (MTV).....	48
8.2.1	Τα Μοντέλα (Models).....	48
8.2.2	Τα Πρότυπα (Templates).....	48
8.2.3	Οι Προβολές (Views).....	49
8.3	Σχεδιασμός Βάσης Δεδομένων.....	49
8.3.1	Διαχείριση Δεδομένων Χρονοσειρών (Time-Series Data) στο IoT.....	50
8.3.2	Serverless Αρχιτεκτονική και Υπολογιστικό Νέφος (Neon PostgreSQL).....	50
8.4	Λογική Backend.....	51
8.4.1	Ο Αλγόριθμος Υπολογισμού SoC (Python Backend).....	51
8.4.2	Ο Αλγόριθμος Αξιολόγησης SoH.....	52
8.4.3	Υπολογισμός Βάθους Εκφόρτισης (DoD) και Εσωτερικής Αντίστασης (IR).....	53

8.5	Ταυτοποίηση και Ασφάλεια	53
Κεφάλαιο 9ο:	Διεπαφή Χρήστη (UI) και Fleet Management	55
9.1	Εισαγωγή.....	55
9.2	Πίνακας Ελέγχου Πελάτη (Client Dashboard).....	55
9.3	Κέντρο Διαχείρισης Στόλου (Fleet Dashboard).....	56
9.4	Σύστημα Συναγερμών (Alerts).....	57
9.5	Διαδραστικότητα και Εμπειρία Χρήστη.....	59
Κεφάλαιο 10ο:	Πειραματικές Δοκιμές και Αποτελέσματα.....	60
10.1	Μεθοδολογία Δοκιμών.....	60
10.2	Αποτελέσματα Κανονικής Λειτουργίας.....	60
10.3	Δοκιμή Συστήματος Προστασίας (LVD).....	61
10.4	Δοκιμή Συναγερμών (Alerts)	61
Κεφάλαιο 11ο:	Συμπεράσματα, Περιορισμοί και Μελλοντικές Επεκτάσεις.....	63
11.1	Συμπεράσματα.....	63
11.2	Περιορισμοί της Έρευνας.....	63
11.3	Προτάσεις για τροποποιήσεις.....	64
11.4	Μελλοντικές Επεκτάσεις.....	64
ΒΙΒΛΙΟΓΡΑΦΙΑ.....		66
ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ		69

Κατάλογος Σχημάτων

Εικόνα 2.1: Η αρχιτεκτονική τριών επιπέδων.....	4
Εικόνα 2.2: Δομή ενός ES.....	6
Εικόνα 3.1: Καμπύλες εκφόρτισης VRLA 12V στους 25°C	11
Εικόνα 3.2: Η φόρτιση τριών σταδίων	12
Σχήμα 4.1: Αρχιτεκτονική του Συστήματος.....	15
Εικόνα 6.1: Ο ESP32 NodeMCU-32S	27
Εικόνα 6.2: Αισθητήρας Θερμοκρασίας DS18B20.....	28
Εικόνα 6.3: Ο αισθητήρας INA226.....	28
Εικόνα 6.4: Τυπική υλοποίηση I2C	29
Εικόνα 6.5: Ο Ελεγκτής Φόρτισης.....	31
Εικόνα 6.6: Λαμπτήρας P21W BA15S 1156	32
Εικόνα 6.7: LM2596	32
Εικόνα 6.8: Το MOSFET	33
Εικόνα 6.9: Εγκατάσταση Fritzing.....	34
Σχέδιο 6.1: Το Ηλεκτρονικό Κύκλωμα	35
Εικόνα 6.10: Σύνδεση μπαταρίας, ρυθμιστή, ασφάλειας και shunt	36
Εικόνα 6.11: Γενική επισκόπηση της δοκιμαστικής διάταξης σε breadboard	37
Εικόνα 6.12: Δοκιμή ESP32 στο breadboard	37
Εικόνα 6.13: Ρύθμιση τροφοδοσίας 5V LM2596	37
Εικόνα 6.14: Συγκόλληση headers στη διάτρητη.....	38
Εικόνα 6.15: Σταθεροποίηση μονάδων στη βάση.....	39
Εικόνα 6.16: Τελική διάταξη βάσης πριν τη δοκιμή.....	39
Εικόνα 6.17: Η τελική μορφή της πειραματικής διάταξης.....	40
Εικόνα 7.1: Συνάρτηση αρχικοποίησης και σύνδεσης στο ασύρματο δίκτυο.....	41
Εικόνα 7.2: Δομή και κωδικοποίηση των τηλεμετρικών δεδομένων (JSON).....	42
Εικόνα 7.3: MQTT έναντι HTTP	43

Εικόνα 7.4: Διαδικασία αποστολής HTTP POST και εκκαθάριση μνήμης	43
Σχήμα 7.1: Διάγραμμα Ροής Προγράμματος ESP32	45
Εικόνα 7.5: Εισαγωγή απαραίτητων βιβλιοθηκών - αρχικοποίηση hardware	46
Εικόνα 7.6: Αλγόριθμος τοπικού ελέγχου για την προστασία της μπαταρίας	47
Εικόνα 8.1: Το αρχιτεκτονικό πρότυπο MTV	48
Εικόνα 8.2: ERD Βάσης Δεδομένων	49
Εικόνα 9.1: Ο πίνακας ελέγχου του πελάτη	55
Εικόνα 9.2: Το αναδυόμενο παράθυρο για κάθε δείκτη	56
Εικόνα 9.3: Ο πίνακας ελέγχου του διαχειριστή	57
Εικόνα 9.4: Τα e-mail προειδοποίησης	58
Εικόνα 10.1: Η πτώση της τάσης	61
Εικόνα 10.2: Αυτοματοποιημένο μήνυμα συναγερμού	62

Κατάλογος Πινάκων

Πίνακας 4.1: Λειτουργικές Απαιτήσεις.....	16
Πίνακας 4.2: Μη Λειτουργικές Απαιτήσεις.....	17
Πίνακας 4.3: Σύγκριση με το Σύστημα των Vyshnav κ.ά. [6]	20
Πίνακας 5.1: Παράδειγμα κώδικα MicroPython.....	21
Πίνακας 5.2: Παράδειγμα κώδικα Python.....	22
Πίνακας 5.3: Παράδειγμα κώδικα HTML – JavaScript – Chart.js.....	23
Πίνακας 6.1: Κοστολόγιο του συστήματος.....	40
Πίνακας 8.1: Υπολογισμός του SoC	51
Πίνακας 8.2: Υπολογισμός του SoH	52
Πίνακας 9.1:Ο αλγόριθμος ελέγχου ορίων και η λογική του cooldown	58

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΙΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
BMS	Battery Management System
DoD	Depth of Discharge
DS18B20	Αισθητήρας Θερμοκρασίας DS18B20
ES	Embedded Systems
ESP32	Υπολογιστική Πλατφόρμα ESP32
GPIO	General Purpose Input/Output
I2C	Το πρωτόκολλο I2C
IoT	Internet of Things
JSON	JavaScript Object Notation
LVD	Low Voltage Disconnect
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MTV	Model, Template, View
OCV	Open Circuit Voltage
ORM	Object-Relational Mapper
PWM	Pulse Width Modulation
RTOS	Real-Time Operating System
REST	Representational State Transfer
SoC	State of Charge
SoH	State of Health
UI	User Interface

Κεφάλαιο 1ο: Εισαγωγή

1.1 Πλαίσιο και κίνητρο

Η παγκόσμια στροφή προς τις Ανανεώσιμες Πηγές Ενέργειας (ΑΠΕ) και ειδικότερα προς τα φωτοβολταϊκά συστήματα αποτελεί θεμέλιο λίθο της σύγχρονης ενεργειακής μετάβασης. Στις αυτόνομες (off-grid) εγκαταστάσεις, η ομαλή λειτουργία των φορτίων εξαρτάται αποκλειστικά από την αξιοπιστία των συστημάτων αποθήκευσης ενέργειας (μπαταρίες). Παρά την άνοδο νέων χημικών συστάσεων, οι συσσωρευτές μολύβδου-οξέος κλειστού τύπου (VRLA) εξακολουθούν να κατέχουν σημαντικό μερίδιο λόγω του χαμηλού αρχικού κόστους κτήσης και της αποδεδειγμένης στιβαρότητάς τους.

Ωστόσο, οι συσσωρευτές αυτοί είναι ιδιαίτερα ευάλωτοι σε φαινόμενα όπως η βαθιά εκφόρτιση, η υπερθέρμανση και η σουλφάτωση, τα οποία οδηγούν σε πρόωρη γήρανση και καταστροφή των στοιχείων τους. Το ισχυρότερο κίνητρο της παρούσας διπλωματικής εργασίας είναι η εκμετάλλευση του Διαδικτύου των Πραγμάτων (Internet of Things - IoT) και των τεχνολογιών Υπολογιστικής στα Άκρα (Edge Computing), ώστε να αναπτυχθεί μια προσιτή, ανοιχτού κώδικα (open-source) και αξιόπιστη λύση που να προστατεύει το ενεργειακό κεφάλαιο και να παρατείνει τη διάρκεια ζωής των συσσωρευτών σε πραγματικό χρόνο.

1.2 Ορισμός του προβλήματος

Στη συντριπτική πλειονότητα των μικρών και μεσαίων αυτόνομων φωτοβολταϊκών εγκαταστάσεων, η επιτήρηση των μπαταριών είναι ελλιπής ή ανύπαρκτη. Οι παραδοσιακοί ελεγκτές φόρτισης προσφέρουν μόνο βασικές ενδείξεις, καθιστώντας δυσχερή την έγκαιρη διάγνωση της εσωτερικής φθοράς.

Τα κύρια προβλήματα που εντοπίζονται συνοψίζονται στα εξής:

- **Απουσία Τηλεμετρίας Υψηλής Ακρίβειας:** Οι χρήστες δεν γνωρίζουν σε πραγματικό χρόνο το πραγματικό επίπεδο φόρτισης (SoC) και την κατάσταση υγείας (SoH) του συσσωρευτή.
- **Μη Αναστρέψιμη Φθορά λόγω Βαθιάς Εκφόρτισης:** Όταν η τάση πέφτει κάτω από τα κρίσιμα όρια, η καθυστερημένη αντίδραση καταστρέφει μόνιμα τις πλάκες μολύβδου.
- **Έλλειψη Κεντρικής Διαχείρισης Στόλου (Fleet Management):** Εταιρείες συντήρησης και διαχειριστές πολλαπλών σταθμών αδυνατούν να εντοπίσουν απομακρυσμένα ποιες μπαταρίες υπολειτουργούν, οδηγώντας σε υψηλό κόστος επιτόπιων επισκέψεων.
- **Υψηλό Κόστος Εμπορικών Λύσεων BMS:** Τα υπάρχοντα βιομηχανικά συστήματα διαχείρισης μπαταριών είναι συχνά κλειστής αρχιτεκτονικής, ασύμβατα με σύγχρονες διαδικτυακές υποδομές και οικονομικά ασύμφορα για μικρές εγκαταστάσεις.

1.3 Διαμόρφωση κειμένου

Η παρούσα εργασία προσεγγίζει το πρόβλημα μέσα από μια ολοκληρωμένη μεθοδολογία σχεδιασμού και υλοποίησης από την πλευρά του μηχανικού (end-to-end engineering approach). Η ανάλυση διαμορφώνεται ακολουθώντας πιστά το διεθνές αρχιτεκτονικό πρότυπο των Τριών Επιπέδων του IoT:

1. **Επίπεδο Αντίληψης (Hardware):** Σχεδιασμός και κατασκευή της φυσικής διάταξης, επιλογή αισθητήρων ακριβείας και ανάπτυξη του κυκλώματος ισχύος/αποκοπής (MOSFET).
2. **Επίπεδο Δικτύου (Network):** Υλοποίηση ευρωστίας επικοινωνίας μέσω MicroPython, δομημένων πακέτων JSON και του πρωτοκόλλου HTTP POST.
3. **Επίπεδο Εφαρμογής (Cloud & Software):** Ανάπτυξη του Django Backend, μοντελοποίηση της βάσης PostgreSQL (Neon Serverless), υπολογισμός των ηλεκτροχημικών δεικτών (SoC, SoH, IR) και σχεδιασμός διαδραστικών διεπαφών χρήστη (Chart.js).

1.4 Στόχοι και ερευνητικά ερωτήματα

Κύριος Στόχος: Ο σχεδιασμός, η υλοποίηση και η πειραματική αξιολόγηση ενός ολοκληρωμένου, χαμηλού κόστους συστήματος IoT σε Python για την παρακολούθηση σε πραγματικό χρόνο και την αυτόνομη προστασία μπαταριών φωτοβολταϊκών.

Ειδικοί Στόχοι:

- Η κατασκευή αξιόπιστου υλικού (Hardware) με χρήση του μικροελεγκτή ESP32 και αισθητήρων INA226 και DS18B20.
- Η ενσωμάτωση αλγορίθμων Τοπικού Συστήματος Διαχείρισης (Edge BMS) για ταχεία αποκοπή φορτίου (LVD) και θερμική προστασία.
- Η ανάπτυξη σύγχρονου Cloud Backend (Django/PostgreSQL) για αποθήκευση, ανάλυση και εκτίμηση δεικτών SoC, SoH και Εσωτερικής Αντίστασης (IR).
- Η δημιουργία εύχρηστου Web Dashboard με διαδραστικά γραφήματα και αυτοματοποιημένο σύστημα ειδοποιήσεων (Alerts via Email).

Ερευνητικά Ερωτήματα (Research Questions):

EE1: Μπορεί ένας μικροελεγκτής χαμηλού κόστους (ESP32) να εκτελέσει αξιόπιστη τοπική προστασία (Edge BMS) ανεξάρτητα από τη διαθεσιμότητα του διαδικτύου;

EE2: Πόσο αποτελεσματική είναι η χρήση της OCV για την αρχική εκτίμηση του SoC σε συνδυασμό με τη μέθοδο Coulomb Counting για τη δυναμική παρακολούθησή του σε συνθήκες λειτουργίας;

EE3: Πώς μπορεί μια ενιαία αρχιτεκτονική λογισμικού σε Python να καλύψει ταυτόχρονα τις ανάγκες του οικιακού χρήστη και του κεντρικού διαχειριστή στόλου εγκαταστάσεων;

1.5 Συνεισφορά

Η συνεισφορά της παρούσας εργασίας έγκειται στα εξής σημεία:

- **Ολοκληρωμένη Full-Stack Python Υλοποίηση:** Ενοποίηση όλων των επιπέδων (από το firmware με MicroPython έως το backend με Django) σε ένα ενιαίο οικοσύστημα λογισμικού.
- **Υπολογιστική στα Άκρα (Edge Computing):** Μεταφορά της κρίσιμης λογικής ασφαλείας απευθείας στον μικροελεγκτή, ελαχιστοποιώντας την καθυστέρηση απόκρισης και αδιάλειπτη προστασία.
- **Δυναμικός Υπολογισμός Εσωτερικής Αντίστασης (IR):** Ανάπτυξη αλγορίθμου εξαγωγής της εσωτερικής αντίστασης σε πραγματικό χρόνο βάσει της πτώσης τάσης υπό φορτίο (Voltage Sag), παρέχοντας έναν πρόσθετο δείκτη για την αξιολόγηση της κατάστασης της μπαταρίας.
- **Πειραματική Τεκμηρίωση σε Πραγματικές Συνθήκες:** Πειραματική επιβεβαίωση της βασικής λειτουργίας του συστήματος μέσω πειράματος επιταχυνόμενης εκφόρτισης (Accelerated Testing) και καταγραφή των αποτελεσμάτων.

1.6 Δομή της εργασίας

Η διπλωματική εργασία είναι οργανωμένη ως εξής:

- **Κεφάλαιο 2ο (Τεχνολογίες IoT, Ενσωματωμένα Συστήματα και Web APIs):** Αναλύει τις βασικές έννοιες, την αρχιτεκτονική του IoT και τις αρχές των REST APIs.
- **Κεφάλαιο 3ο (Θεωρητικό Υπόβαθρο Συσσωρευτών):** Παρουσιάζει τη χημεία των μπαταριών VRLA, τα στάδια φόρτισης και τις μεθοδολογίες εκτίμησης SoC και SoH.

- **Κεφάλαιο 4ο (Συνολική Αρχιτεκτονική του Συστήματος):** Περιγράφει τη γενική δομή, τα σενάρια χρήσης και τη ροή της πληροφορίας.
- **Κεφάλαιο 5ο (Εργαλεία Ανάπτυξης και Γλώσσες Προγραμματισμού):** Παρουσιάζει τα εργαλεία λογισμικού (MicroPython, Python, Django, PostgreSQL, VS Code).
- **Κεφάλαιο 6ο (Επίπεδο Αντίληψης - Hardware):** Αναλύει τα επιμέρους ηλεκτρονικά εξαρτήματα, τη συνδεσμολογία και τη φυσική κατασκευή.
- **Κεφάλαιο 7ο (Επίπεδο Δικτύου):** Εστιάζει στη δικτύωση Wi-Fi, στο πρωτόκολλο HTTP POST και στο διάγραμμα ροής του ESP32.
- **Κεφάλαιο 8ο (Επίπεδο Εφαρμογής):** Αναλύει τη βάση δεδομένων, το Django Backend και τους αλγορίθμους υπολογισμού SoC/SoH.
- **Κεφάλαιο 9ο (Διεπαφή Χρήστη και Fleet Management):** Παρουσιάζει το σχεδιασμό των dashboards και το σύστημα συναγερμών (Email Alerts).
- **Κεφάλαιο 10ο (Πειραματικές Δοκιμές και Αποτελέσματα):** Καταγράφει τα αποτελέσματα των πειραμάτων εκφόρτισης και την αποκριση του συστήματος προστασίας.
- **Κεφάλαιο 11ο (Συμπεράσματα και Μελλοντικές Επεκτάσεις):** Συνοψίζει τα ευρήματα της εργασίας και προτείνει κατευθύνσεις για μελλοντική έρευνα.

1.7 Επίλογος

Στο παρόν κεφάλαιο τέθηκαν τα θεμέλια, το κίνητρο και οι στόχοι της διπλωματικής εργασίας, αναδεικνύοντας τη σημασία της επιτήρησης των συστημάτων αποθήκευσης ενέργειας. Στο επόμενο κεφάλαιο αναλύονται οι βασικές τεχνολογίες του Διαδικτύου των Πραγμάτων (IoT), των Ενσωματωμένων Συστημάτων και των Web APIs που αξιοποιήθηκαν για την υλοποίηση της προτεινόμενης λύσης.

Κεφάλαιο 2ο: Τεχνολογίες του IoT: Ενσωματωμένα Συστήματα και Web APIs

2.1 Εισαγωγή

Το Διαδίκτυο των Πραγμάτων (IoT) έχει αλλάξει ριζικά την καθημερινότητά μας, παρέχοντας τη δυνατότητα απομακρυσμένης παρακολούθησης και ελέγχου των «έξυπνων» συσκευών που μας περιβάλλουν μέσω του διαδικτύου. Η παρούσα εργασία πραγματεύεται τον Σχεδιασμό και την Ανάπτυξη ενός Συστήματος IoT, υλοποιημένου σε Python, με σκοπό την παρακολούθηση σε πραγματικό χρόνο της Κατάστασης Υγείας (State of Health - SoH) και της Κατάστασης Φόρτισης (State of Charge - SoC) μπαταριών φωτοβολταϊκών συστημάτων. Το έργο αυτό επιτυγχάνεται συνδυάζοντας τεχνολογίες αιχμής από τους τομείς των:

- Ενσωματωμένων Συστημάτων (Embedded Systems)
- Web APIs

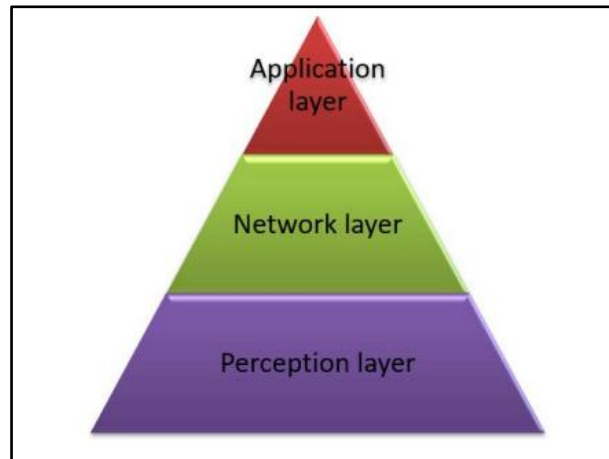
Στο παρόν κεφάλαιο παρουσιάζεται το απαραίτητο τεχνολογικό υπόβαθρο πάνω στο οποίο βασίζεται ο σχεδιασμός και η υλοποίηση του προτεινόμενου συστήματος. Αρχικά, στην Ενότητα 2.2 εξετάζονται οι βασικές έννοιες του Διαδικτύου των Πραγμάτων (IoT), αναλύεται η αρχιτεκτονική των τριών επιπέδων και καταγράφονται τα κύρια χαρακτηριστικά των σύγχρονων IoT εφαρμογών. Στη συνέχεια, στην Ενότητα 2.3 προσεγγίζονται τα Ενσωματωμένα Συστήματα (Embedded Systems), περιγράφεται η δομή τους και αναλύονται οι ιδιαίτεροι περιορισμοί υλικού και επεξεργαστικής ισχύος που διέπουν τη λειτουργία τους σε πραγματικό χρόνο. Τέλος, στην Ενότητα 2.4 παρουσιάζεται η αρχιτεκτονική και η φιλοσοφία λειτουργίας των Web APIs, τα πρωτόκολλα επικοινωνίας (HTTP, REST) και τα πρότυπα ανταλλαγής δεδομένων (JSON), αναδεικνύοντας τον κρίσιμο ρόλο τους στην απρόσκοπτη διασύνδεση των τερματικών συσκευών με τις εφαρμογές διαδικτύου.

2.2 Διαδίκτυο των Πραγμάτων (IoT)

Το Διαδίκτυο των Πραγμάτων θα μπορούσε να οριστεί ως ένα δίκτυο φυσικών αντικειμένων όπως για παράδειγμα συσκευές, οχήματα, κτίρια, τα οποία έχουν ενσωματωμένα ηλεκτρονικά συστήματα, κυκλώματα, λογισμικό, αισθητήρες και διαθέτουν συνδεσιμότητα δικτύου. Αυτά τα στοιχεία επιτρέπουν στα αντικείμενα να ανταλλάσσουν και να συλλέγουν δεδομένα, με σταθερές αναβαθμίσεις σε πολλούς τομείς [1].

2.2.1 Αρχιτεκτονική του IoT

Η αρχιτεκτονική των εφαρμογών του IoT μπορεί να παρουσιαστεί με πολλούς τρόπους. Μια αρχιτεκτονική είναι αυτή των τριών επιπέδων που φαίνεται στην Εικόνα 2.1.



Εικόνα 2.1: Η αρχιτεκτονική τριών επιπέδων [2]

Τα επίπεδα που δείχνει η παραπάνω εικόνα αναφέρονται στα παρακάτω επίπεδα:

- **Επίπεδο Αντίληψης (Perception Layer):** Αυτό το επίπεδο αποτελείται από μικρές, φθηνές και ισχυρές συσκευές που είναι ικανές να ανιχνεύουν, να επεξεργάζονται και να επικοινωνούν καθώς και από μια ποικιλία αισθητήρων [35].
- **Επίπεδο Δικτύου (Network Layer):** Το Επίπεδο Δικτύου είναι υπεύθυνο για τη μετάδοση των δεδομένων από το επίπεδο αντίληψης προς το ανώτερο επίπεδο, καθώς και για τη λήψη δεδομένων από αυτό [35].
- **Επίπεδο Εφαρμογής (Application Layer):** Το Επίπεδο Εφαρμογής παρέχει στον τελικό χρήστη εξατομικευμένες εφαρμογές για την αποθήκευση, την επεξεργασία και την ανάλυση των συλλεχθέντων δεδομένων. Μέσω αυτού επιτυγχάνεται η παρακολούθηση πληροφοριών σε πραγματικό χρόνο, καθώς και ο απομακρυσμένος έλεγχος με τη βοήθεια εργαλείων οπτικοποίησης. Παράλληλα, αξιοποιώντας τεχνικές μηχανικής μάθησης το σύστημα μπορεί να λαμβάνει αυτόματα έξυπνες και έγκαιρες αποφάσεις, προλαμβάνοντας αποτελεσματικά τις ζημιές [35].

2.2.2 Χαρακτηριστικά του IoT

Οι IoT εφαρμογές έχουν τα παρακάτω βασικά χαρακτηριστικά [3]:

- **Διασυνδεσιμότητα:** Η διασυνδεσιμότητα είναι απαραίτητη καθώς τα αντικείμενα πρέπει να είναι συνεχώς συνδεδεμένα μεταξύ τους. Δημιουργεί ένα παγκόσμιο δίκτυο όπου φυσικά και εικονικά αντικείμενα ενσωματώνονται ως δίκτυο πληροφοριών και μπορούν να ανταλλάσσουν δεδομένα.
- **Ετερογένεια:** Οι συσκευές στο IoT είναι ετερογενείς, καθώς βασίζονται σε διαφορετικές πλατφόρμες υλικού και δικτύα. Μπορούν να αλληλεπιδρούν με άλλες συσκευές ή πλατφόρμες υπηρεσιών μέσω διαφορετικών δικτύων.
- **Δυναμική προσαρμοστικότητα:** Η κατάσταση των συσκευών αλλάζει σε πραγματικό χρόνο, για αυτό κρίνεται απαραίτητο η αρχιτεκτονική να είναι προσαρμοστική, επιτρέποντας στις συσκευές να αλληλεπιδρούν δυναμικά.
- **Επεκτασιμότητα:** Ο όρος αυτός περιγράφει την ευελιξία και την επεκτασιμότητα ενός συστήματος IoT, δηλαδή τη δυνατότητά του να αναβαθμίζεται διαρκώς προκειμένου να συμβαδίζει με τις μελλοντικές τεχνολογικές εξελίξεις.
- **Ασφάλεια:** Καθώς επωφελούμαστε από το IoT, δεν πρέπει να ξεχνάμε την ασφάλεια. Η ασφάλεια πρέπει να διαχειρίζεται κυρίως στο Επίπεδο Δικτύου, χρησιμοποιώντας κατάλληλες τεχνολογίες για την προστασία των δεδομένων, καθώς το IoT συνδέει πολλά προσωπικά αντικείμενα, δημιουργώντας δυνητικούς κινδύνους.

- **Συνδεσιμότητα:** Αφορά τις υποδομές που υποστηρίζουν τις ασύρματες ή ενσύρματες συνδέσεις μεταξύ των συσκευών. Είναι απαραίτητη προκειμένου οι αισθητήρες να μπορούν να επικοινωνούν (communicate) και τα δεδομένα τους να είναι προσβάσιμα και ανακτήσιμα.

Το Διαδίκτυο των Πραγμάτων έχει παρουσιάσει ραγδαία ανάπτυξη τα τελευταία χρόνια, με τον αριθμό των «έξυπνων» συσκευών να αυξάνεται από περίπου 11 δισεκατομμύρια το 2020 σε σχεδόν 20 δισεκατομμύρια σήμερα (2026), ενώ προβλέπεται να ξεπεράσει τα 40 δισεκατομμύρια μέσα στην επόμενη δεκαετία.

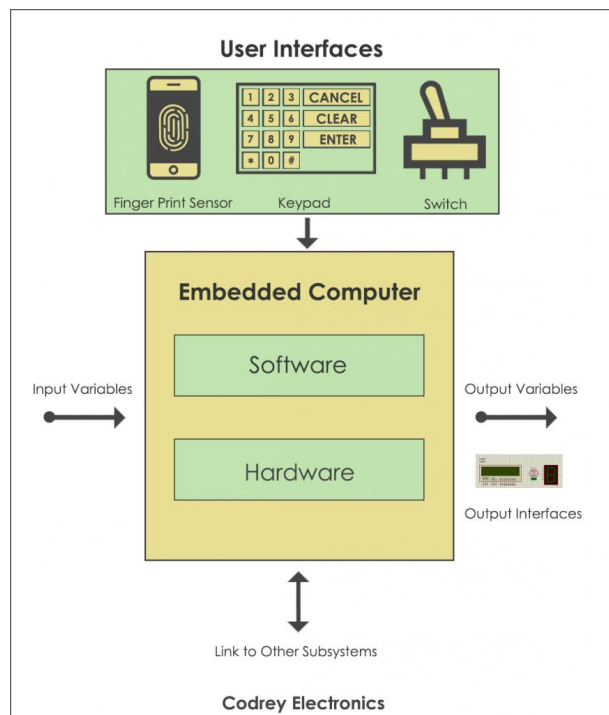
2.3 Ενσωματωμένα συστήματα

Τα ενσωματωμένα συστήματα (Embedded Systems) είναι υπολογιστικές μονάδες ειδικού σκοπού, σχεδιασμένες να αναλαμβάνουν συγκεκριμένες διεργασίες ως αναπόσπαστα μέρη μεγαλύτερων ηλεκτρομηχανικών διατάξεων. Βασικά χαρακτηριστικά τους αποτελούν η λειτουργία σε πραγματικό χρόνο και οι αυστηροί περιορισμοί σε επεξεργαστική ικανότητα, διαθέσιμη μνήμη και ενεργειακή κατανάλωση. Ένας άλλος σχετικός όρος είναι τα CPS (Cyber-Physical Systems) που επεκτείνει τα ES για να λάβει επιπλέον υπόψη το φυσικό περιβάλλον, κατά τη διάρκεια του σχεδιασμού του συστήματος [4].

Το Internet of Things (IoT) αναβαθμίζει αυτά τα συστήματα, ενσωματώνοντας δυνατότητες δικτυακής διασύνδεσης και απομακρυσμένου ελέγχου. Ένα ενσωματωμένο σύστημα IoT αξιοποιεί μικροελεγκτές, αισθητήρες, ενεργοποιητές (actuators) και εξαρτήματα επικοινωνίας προκειμένου να αντλεί δεδομένα από τον φυσικό κόσμο, να τα επεξεργάζεται τοπικά και να τα μεταδίδει σε κεντρικούς διακομιστές (servers).

2.3.1 Επισκόπηση ES

Η δομή των ES φαίνεται στην Εικόνα 2.2



Εικόνα 2.2: Δομή ενός ES [8]

Σε ένα ενσωματωμένο σύστημα υπάρχει συνδυασμένη χρήση του υλικού ES και του λογισμικού ES και ο κεντρικός πυρήνας επεξεργασίας είναι αυτός που αλληλοεπιδρά με τον εξωτερικό κόσμο. Πιο αναλυτικά, η αρχιτεκτονική του χωρίζεται στα εξής βασικά μέρη:

Ο Πυρήνας: Είναι η "καρδιά" του συστήματος και αποτελείται από:

- **Υλικό (Hardware):** Περιλαμβάνει τον μικροεπεξεργαστή ή μικροελεγκτή, τη μνήμη (RAM/ROM) και τα κυκλώματα ελέγχου.
- **Λογισμικό (Software):** Το πρόγραμμα που καθοδηγεί το υλικό για να εκτελέσει τη συγκεκριμένη εργασία, συχνά με τη βοήθεια ενός Λειτουργικού Συστήματος Πραγματικού Χρόνου (RTOS).
- **Διεπαφές Εισόδου (Input Variables & User Interfaces):** Είναι τα μέσα με τα οποία το σύστημα λαμβάνει δεδομένα ή εντολές. Περιλαμβάνουν διάφορους αισθητήρες όπως αισθητήρες υγρασίας ή θερμοκρασίας, διακόπτες και άλλες μεταβλητές εισόδου από το περιβάλλον [9].
- **Διεπαφές Εξόδου (Output Variables & Output Interfaces):** Αφού το σύστημα επεξεργαστεί τα δεδομένα, παράγει αποτελέσματα. Αυτά μπορεί να εμφανίζονται σε οθόνες/ενδείξεις όπως σε μια οθόνη LCD ή να είναι σήματα ελέγχου προς ενεργοποιητές (π.χ. μοτέρ, βαλβίδες) [9].
- **Σύνδεση με άλλα Υποσυστήματα (Link to Other Subsystems):** Πολλά σύγχρονα ενσωματωμένα συστήματα δεν λειτουργούν απομονωμένα, αλλά επικοινωνούν με άλλα συστήματα, αισθητήρες ή δίκτυα μέσω Wi-Fi, Bluetooth, ή καλωδιακών πρωτοκόλλων επικοινωνίας.
- **Λειτουργικό Σύστημα (RTOS):** Το RTOS είναι υπεύθυνο για τον συντονισμό των εργασιών, τη διαχείριση της μνήμης και, κυρίως, την εξασφάλιση ότι κρίσιμες διεργασίες θα εκτελεστούν μέσα σε αυστηρά χρονικά όρια.
- **Λογισμικό Εφαρμογής:** Είναι το ανώτερο επίπεδο λογισμικού και αφορά τον τελικό κώδικα που καθορίζει τη λειτουργικότητα της συσκευής. Είναι ουσιαστικά το πρόγραμμα που παίρνει τις αποφάσεις.

Συμπερασματικά, η γενική δομή ενός ενσωματωμένου συστήματος βελτιστοποιείται πάντα με αποκλειστικό γνώμονα την εφαρμογή για την οποία προορίζεται. Αυτός ο εξειδικευμένος σχεδιασμός επιβάλλει συγκεκριμένους περιορισμούς και ιδιαιτερότητες, τα οποία αναλύονται στην επόμενη ενότητα ως τα βασικά του χαρακτηριστικά.

2.3.2 Χαρακτηριστικά ES

Τα κυριότερα χαρακτηριστικά των ES είναι:

- **Αυστηρές Μετρικές Σχεδιασμού:** Ο σχεδιασμός τους υπαγορεύεται από πολύ αυστηρούς περιορισμούς, οι οποίοι αποτελούν μέτρο της απόδοσης τους, όπως το μέγεθος, το κόστος κατασκευής, η κατανάλωση ισχύος και η ταχύτητα.
- **Διαχείριση Κατάστασης Λειτουργίας:** Για να διατηρείται η κατανάλωση ενέργειας σε χαμηλά επίπεδα, τα συστήματα αυτά πρέπει να διαχειρίζονται αποτελεσματικά την κατανάλωση ισχύος, χρησιμοποιώντας διαφορετικές καταστάσεις λειτουργίας.
- **Διαβαθμισμένη Αξιοπιστία και Σταθερότητα:** Οι απαιτήσεις ποιότητας και αξιοπιστίας διαφέρουν δραματικά ανάλογα με την εφαρμογή. Για παράδειγμα, μια βλάβη σε μια ιατρική συσκευή κατά τη διάρκεια μιας εγχείρησης μπορεί να αποβεί μοιραία, ενώ μια βλάβη στο τηλεχειριστήριο μιας τηλεόρασης αποτελεί απλώς μια ταλαιπωρία. Επίσης αναμένεται να λειτουργούν συνεχώς για χρόνια χωρίς σφάλματα και συχνά κάτω από αντίξοες συνθήκες (ακραίες θερμοκρασίες, δονήσεις, υγρασία).
- **Εκτέλεση Συγκεκριμένης Λειτουργίας:** Έχουν σχεδιαστεί για να εκτελούν μία ή μερικές συγκεκριμένες εργασίες (π.χ. ο έλεγχος των φρένων ABS σε ένα αυτοκίνητο) [10].
- **Λειτουργία σε Πραγματικό Χρόνο:** Πολλά από αυτά είναι "Real-Time Systems", που σημαίνει ότι πρέπει να αντιδρούν σε ερεθίσματα και να ολοκληρώνουν εργασίες μέσα σε

αυστηρά, προκαθορισμένα χρονικά όρια (π.χ. ο αερόσακος πρέπει να ανοίξει σε χιλιοστά του δευτερολέπτου).

- **Περιορισμένοι Πόροι:** Διαθέτουν περιορισμένη επεξεργαστική ισχύ, μικρή χωρητικότητα μνήμης και ελάχιστο αποθηκευτικό χώρο, προκειμένου να διατηρείται το μέγεθος και το κόστος χαμηλά.
- **Χαμηλή Κατανάλωση Ενέργειας:** Επειδή πολλά συστήματα (όπως συσκευές IoT ή wearables) λειτουργούν με μπαταρίες, η αρχιτεκτονική τους βελτιστοποιείται ώστε να καταναλώνουν την ελάχιστη δυνατή ενέργεια.

2.3.3 Εφαρμογές Μικροελεγκτών σε Συστήματα IoT

Οι Dey και Bera [5] παρουσίασαν την ανάπτυξη μιας συμπαγούς συσκευής IoT βασισμένης σε ESP32, ικανής να μετρά ατμοσφαιρική πίεση, θερμοκρασία και υγρασία, να καταγράφει καρδιακό ρυθμό μέσω σύζευξης Bluetooth με εφαρμογή Android, και να επαναπρογραμματίζεται για διαφορετικές χρήσεις, από οικιακό αυτοματισμό έως έλεγχο drone. Η εργασία αυτή επιβεβαιώνει την ευελιξία του ESP32 ως πλατφόρμας γενικού σκοπού για IoT εφαρμογές, χωρίς ωστόσο να εστιάζει σε κάποιο συγκεκριμένο πεδίο εφαρμογής υψηλής κρισιμότητας. Η παρούσα εργασία αξιοποιεί την ίδια βασική πλατφόρμα (ESP32), εξειδικεύοντάς την όμως σε ένα σενάριο όπου η αξιοπιστία και ο χρόνος αποκρίσης έχουν άμεσο αντίκτυπο στη διάρκεια ζωής ενός φυσικού συστήματος (του συσσωρευτή), και όχι απλώς στην καταγραφή περιβαλλοντικών δεδομένων.

2.4 WEB APIs

Το API είναι ένα σύνολο από κανόνες και πρωτόκολλα που επιτρέπουν σε ένα λογισμικό να επικοινωνεί με ένα άλλο. Στην περίπτωση των Web APIs, αυτή η επικοινωνία γίνεται μέσω του διαδικτύου μεταξύ διακομιστή (server) και ενός πελάτη (client), ο οποίος μπορεί να είναι ένας περιηγητής ιστού (browser) ή –όπως στην παρούσα εργασία– μια ενσωματωμένη συσκευή IoT.

2.4.1 Αρχιτεκτονική και Λειτουργία Web APIs

- **Το Μοντέλο Client - Server (Πελάτη - Διακομιστή):** Η αρχιτεκτονική των Web APIs βασίζεται αυστηρά σε αυτό το μοντέλο. Ο Client (Πελάτης) είναι η συσκευή ή η εφαρμογή που ζητά μια πληροφορία (π.χ. ένας browser, ένα κινητό ή ένας μικροελεγκτής). Στη συγκεκριμένη αρχιτεκτονική, ο "Πελάτης" (στην περίπτωσή μας ο μικροελεγκτής ESP32) υποβάλλει ένα αίτημα (Request), ενώ ο "Διακομιστής" (το κεντρικό σύστημα) λαμβάνει το αίτημα, το επεξεργάζεται και επιστρέφει την κατάλληλη απάντηση (Response).
- **Βασική Λειτουργία:** Η διαδικασία είναι κυκλική και συγχρονισμένη. Ο Client δημιουργεί ένα αίτημα, το οποίο δρομολογείται μέσω δικτύου στον Server. Ο Server το αυθεντικοποιεί, τρέχει την αντίστοιχη λογική στον κώδικά του και τελικά συντάσσει μια δομημένη απάντηση την οποία στέλνει πίσω στον Client.
- **REST:** Η πλειοψηφία των σύγχρονων Web APIs, συμπεριλαμβανομένου και αυτού που αναπτύχθηκε για την παρούσα εργασία, ακολουθεί τις αρχές της αρχιτεκτονικής REST (Representational State Transfer). Ένα από τα βασικότερα χαρακτηριστικά του REST είναι η απουσία κατάστασης (statelessness) [11]. Αυτό σημαίνει ότι κάθε αίτημα που αποστέλλεται από τον πελάτη προς τον διακομιστή πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες για την κατανόησή του, καθώς ο διακομιστής δεν αποθηκεύει δεδομένα από προηγούμενες επικοινωνίες (sessions).

2.4.2 Πρωτόκολλα και Μορφές Δεδομένων

- **Πρωτόκολλο HTTP και Μέθοδοι:** Η επικοινωνία βασίζεται κυρίως στο πρωτόκολλο HTTP/HTTPS. Ο Client καθορίζει την πρόθεσή του μέσω συγκεκριμένων μεθόδων: τη μέθοδο GET για την ανάκτηση δεδομένων και τη μέθοδο POST για την αποστολή/αποθήκευση νέων δεδομένων στον Server.
- **Η Μορφή JSON:** Η επικρατέστερη μορφή ανταλλαγής δεδομένων είναι το JSON (JavaScript Object Notation). Είναι ένα ελαφρύ format βασισμένο σε κείμενο, οργανωμένο σε ζεύγη «κλειδιού-τιμής» (key-value), το οποίο είναι εξαιρετικά αποδοτικό για αποστολή μέσω δικτύου.

2.4.3 Ο Ρόλος των Web APIs στα Συστήματα IoT

Στο οικοσύστημα του διαδικτύου των πραγμάτων, οι τερματικές συσκευές συχνά διαθέτουν περιορισμένους πόρους μνήμης και επεξεργαστικής ισχύος. Η χρήση ενός Web API επιτρέπει τον διαχωρισμό των αρμοδιοτήτων: οι συσκευές αναλαμβάνουν μόνο τη λήψη των μετρήσεων και την αποστολή τους (μέσω μορφής JSON), μεταφέροντας όλο το υπολογιστικό βάρος της επεξεργασίας στο ασφαλές περιβάλλον του κεντρικού διακομιστή (Backend) [12]. Αυτή η αρχιτεκτονική εξασφαλίζει την επεκτασιμότητα του συστήματος, επιτρέποντας σε εκατοντάδες αισθητήρες να επικοινωνούν ταυτόχρονα με το ίδιο σύστημα.

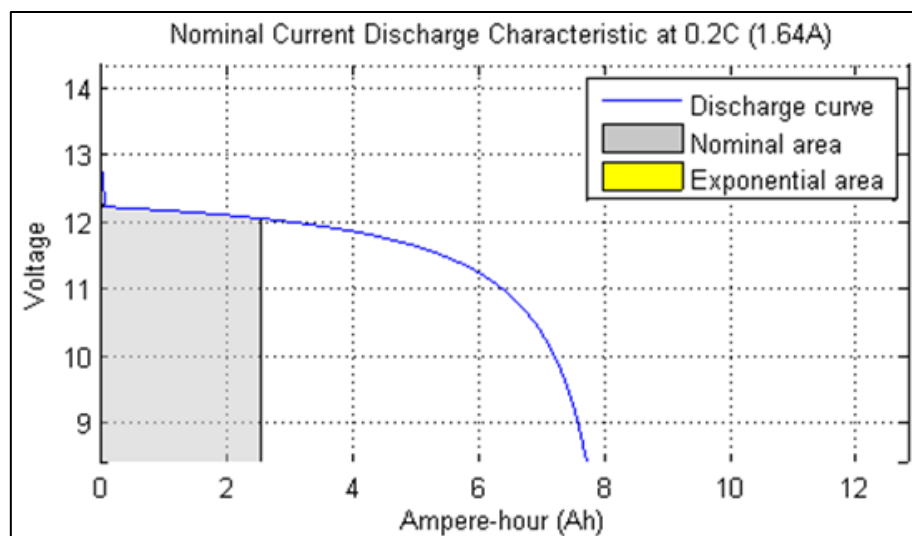
Κεφάλαιο 3ο: Θεωρητικό Υπόβαθρο Συσσωρευτών

3.1 Τεχνολογία Μπαταριών Μολύβδου-Οξέος (VRLA)

Οι συσσωρευτές μολύβδου-οξέος αποτελούν την παλαιότερη και πλέον καθιερωμένη τεχνολογία επαναφορτιζόμενων μπαταριών, έχοντας ευρεία εφαρμογή σε αυτόνομα φωτοβολταϊκά συστήματα, λόγω της αξιοπιστίας τους και του χαμηλού κόστους κτήσης. Στη σύγχρονη εκδοχή τους, κυριαρχούν οι μπαταρίες κλειστού τύπου VRLA (Valve-Regulated Lead-Acid), στις οποίες ανήκει και η μπαταρία που χρησιμοποιείται στην παρούσα κατασκευή.

Η λειτουργία τους βασίζεται σε μια ηλεκτροχημική αντίδραση μεταξύ του διοξειδίου του μολύβδου (θετικός πόλος) και του σπογγώδους μολύβδου (αρνητικός πόλος), τα οποία βρίσκονται εμβαπτισμένα σε ηλεκτρολύτη θεικού οξέος. Το κύριο πλεονέκτημα των μπαταριών VRLA είναι ότι δεν απαιτούν συντήρηση, καθώς διαθέτουν μια βαλβίδα ανακούφισης πίεσης που αποτρέπει τη διαρροή αερίων κατά την κανονική λειτουργία, επιτρέποντας τον ανασυνδυασμό του οξυγόνου και του υδρογόνου στο εσωτερικό τους [25].

Ωστόσο, η συγκεκριμένη τεχνολογία παρουσιάζει και περιορισμούς. Οι μπαταρίες VRLA έχουν χαμηλή ενεργειακή πυκνότητα σε σχέση με τις μπαταρίες λιθίου, είναι ευαίσθητες σε ακραίες θερμοκρασίες και υφίστανται ταχεία υποβάθμιση εάν εκφορτιστούν σε πολύ χαμηλά επίπεδα τάσης. Η τυπική τάση ενός κελιού μολύβδου-οξέος είναι 2V, συνεπώς μια μπαταρία των 12V απαρτίζεται από έξι τέτοια κελιά συνδεδεμένα εν σειρά. Όπως αποτυπώνεται χαρακτηριστικά στην Εικόνα 3.1, η πτώση της τάσης κατά την εκφόρτιση μιας τέτοιας μπαταρίας δεν είναι γραμμική, ενώ η ταχύτητα βύθισής της εξαρτάται άμεσα από το ρεύμα που αντλεί το εκάστοτε φορτίο.



Εικόνα 3.1: Καμπύλες εκφόρτισης VRLA 12V στους 25°C [24]

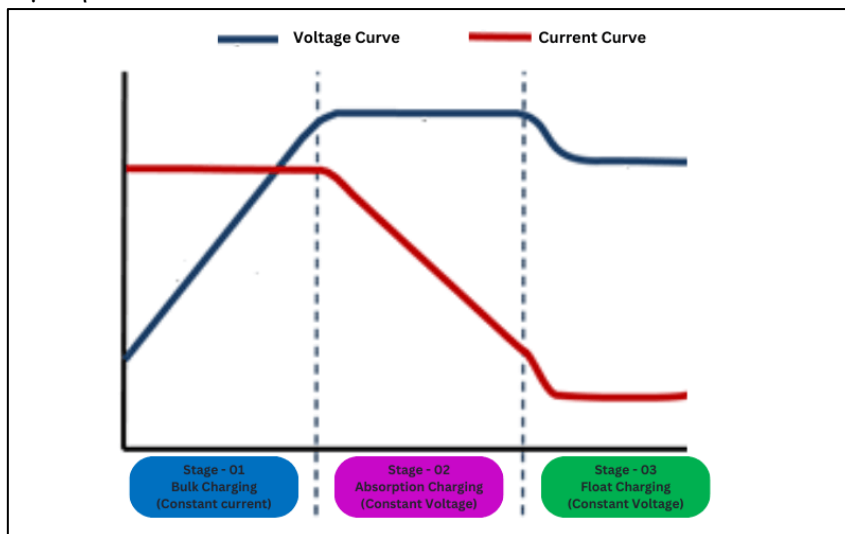
3.2 Στάδια Φόρτισης Φωτοβολταϊκών Συστημάτων

Για τη διασφάλιση της μακροζωίας των συσσωρευτών μολύβδου σε φωτοβολταϊκές εγκαταστάσεις, η διαδικασία αναπλήρωσης της ενέργειας δεν είναι μια απλή, γραμμική διαδικασία. Απαιτείται η χρήση ελεγκτών φόρτισης, οι οποίοι ακολουθούν έναν αυστηρό αλγόριθμο φόρτισης τριών σταδίων του οποίου το διάγραμμα φαίνεται στην Εικόνα 3.2:

1. **Στάδιο Κύριας Φόρτισης (Bulk Charge):** Σε αυτή τη φάση, ο ελεγκτής επιτρέπει τη μέγιστη δυνατή ροή ρεύματος (Constant Current) από το φωτοβολταϊκό πάνελ προς την μπαταρία. Η

τάση της μπαταρίας αυξάνεται σταδιακά μέχρι να φτάσει στο προκαθορισμένο όριο (συνήθως γύρω στα 14.4V - 14.7V). Στο τέλος αυτού του σταδίου, η μπαταρία έχει αναπληρώσει περίπου το 80% της χωρητικότητας της.

2. **Στάδιο Φόρτισης Απορρόφησης (Absorption Charge):** Μόλις η τάση φτάσει στο μέγιστο επιτρεπτό όριο, ο ελεγκτής διατηρεί την τάση σταθερή (Constant Voltage). Καθώς η μπαταρία γεμίζει, η εσωτερική της αντίσταση αυξάνεται και, κατά συνέπεια, το ρεύμα φόρτισης μειώνεται εκθετικά. Το στάδιο αυτό διασφαλίζει το πλήρες και ασφαλές γέμισμα της μπαταρίας, αποτρέποντας την υπερθέρμανση.
3. **Στάδιο Φόρτισης Συντήρησης (Float Charge):** Όταν η μπαταρία φορτιστεί πλήρως, η τάση μειώνεται σε ένα χαμηλότερο, ασφαλές επίπεδο. Ο ελεγκτής παρέχει πλέον μόνο ένα ελάχιστο ρεύμα ικανό να αντισταθμίσει τον φυσικό ρυθμό αυτοεκφόρτισης της μπαταρίας, διατηρώντας την σε ετοιμότητα.



Εικόνα 3.2: Η φόρτιση τριών σταδίων [23]

3.3 Μεθοδολογίες Εκτίμησης Κατάστασης Φόρτισης (SoC)

Η εκτίμηση της Κατάστασης Φόρτισης (State of Charge - SoC) αποτελεί την πιο κρίσιμη λειτουργία ενός Συστήματος Διαχείρισης Μπαταρίας (BMS), καθώς εκφράζει την υπολειπόμενη χωρητικότητα της μπαταρίας ως ποσοστό της ονομαστικής της χωρητικότητας (0% έως 100%). Υπάρχουν διάφορες προσεγγίσεις για τον υπολογισμό του SoC, με τις κυριότερες να αναλύονται παρακάτω:

A. Μέθοδος Τάσης Ανοικτού Κυκλώματος (Open Circuit Voltage - OCV)

Η μέθοδος αυτή βασίζεται στην άμεση σχέση μεταξύ της υπολειπόμενης ενέργειας μιας μπαταρίας και της τάσης της στους ακροδέκτες, όταν αυτή δεν τροφοδοτεί κανένα φορτίο. Για παράδειγμα, μια μπαταρία VRLA 12V θεωρείται 100% φορτισμένη όταν η OCV είναι 12.7V και 0% φορτισμένη όταν πέσει στα 11.5V. Το κύριο πλεονέκτημα της μεθόδου είναι η απλότητα υλοποίησης. Ωστόσο, για να είναι ακριβής, η μπαταρία πρέπει να βρίσκεται σε κατάσταση "ηρεμίας" για αρκετές ώρες, κάτι που σε πραγματικά, δυναμικά συστήματα IoT είναι σπάνιο, καθώς η τάση υπό φορτίο παρουσιάζει πτώση (voltage sag). Επίσης, η καμπύλη τάσης/εκφόρτισης δεν είναι απόλυτα γραμμική.

B. Καταμέτρηση Φορτίου (Coulomb Counting)

Η μέθοδος Coulomb Counting, ή αλλιώς μέθοδος ολοκλήρωσης ρεύματος, καταγράφει με ακρίβεια το ρεύμα που εισέρχεται και εξέρχεται από την μπαταρία στον χρόνο. Μαθηματικά, το SoC υπολογίζεται από την σχέση (3.1):

$$SoC(t) = SoC(t - 1) + [(I(t) * \Delta t) / C] * 100 \quad (3.1)$$

όπου I το ρεύμα και C η ονομαστική χωρητικότητα. Το πλεονέκτημά της είναι ότι προσφέρει δυναμική εκτίμηση σε πραγματικό χρόνο. Το βασικό της μειονέκτημα είναι το φαινόμενο της αθροιστικής συσσώρευσης σφαλμάτων (drift error): εάν η αρχική μέτρηση δεν είναι ακριβής ή ο αισθητήρας ρεύματος έχει μικρή απόκλιση, το σφάλμα αυξάνεται όσο περνάει ο χρόνος. Στη σύγχρονη βιβλιογραφία, τα αποδοτικότερα συστήματα (όπως και η προσέγγιση που επιχειρείται στο λογισμικό της παρούσας εργασίας) χρησιμοποιούν υβριδικές μεθόδους. Συνδυάζουν τη δυναμική συμπεριφορά του Coulomb Counting με περιοδικές διορθώσεις (calibration) βάσει της μεθόδου OCV, όταν το σύστημα ανιχνεύσει ότι η μπαταρία βρίσκεται σε ηρεμία.

Στη σύγχρονη βιβλιογραφία, τα αποδοτικότερα συστήματα χρησιμοποιούν υβριδικές μεθόδους ή προηγμένους αλγορίθμους, όπως τα Φίλτρα Kalman, για την ακριβή εκτίμηση του SoC σε πραγματικό χρόνο, ξεπερνώντας τα προβλήματα συσσώρευσης σφαλμάτων της μεθόδου Coulomb Counting και τη δυσκολία άμεσης μέτρησης της OCV [26].

3.4 Παράγοντες Υποβάθμισης (SoH)

Η Κατάσταση Υγείας (State of Health - SoH) συγκρίνει την τρέχουσα μέγιστη ικανότητα αποθήκευσης ενέργειας της μπαταρίας με τη χωρητικότητα που είχε όταν ήταν καινούργια. Η μείωση του SoH είναι μια μη αναστρέψιμη διαδικασία και επιταχύνεται από συγκεκριμένους παράγοντες:

- **Βάθος Εκφόρτισης (Depth of Discharge - DoD):** Το DoD ορίζεται ως το ποσοστό της αφαιρεθείσας ενέργειας. Οι μπαταρίες μολύβδου υποφέρουν δραματικά όταν εκφορτίζονται πλήρως. Μια μπαταρία VRLA που εκφορτίζεται σταθερά κατά 30% (DoD = 30%) μπορεί να αντέξει πάνω από 1200 κύκλους φόρτισης, ενώ εάν εκφορτίζεται κατά 100% (DoD = 100%) η διάρκεια ζωής της μπορεί να μειωθεί στους μόλις 200 κύκλους. Γι' αυτόν τον λόγο, στην παρούσα εφαρμογή ενσωματώθηκε μηχανισμός LVD (Low Voltage Disconnect) στα 11.5V.
- **Θερμοκρασία (Temperature):** Οι μπαταρίες είναι σχεδιασμένες να αποδίδουν ιδανικά στους 25°C. Σύμφωνα με τον κανόνα του Arrhenius για τις χημικές αντιδράσεις, για κάθε 10°C αύξηση της θερμοκρασίας περιβάλλοντος πάνω από τους 25°C, η αναμενόμενη διάρκεια ζωής της μπαταρίας μολύβδου μειώνεται στο μισό. Σε πολύ υψηλές θερμοκρασίες παρατηρείται απώλεια υγρών και διάβρωση των πλακών.
- **Σουλφάτωση (Sulfation):** Όταν μια μπαταρία μολύβδου παραμένει σε κατάσταση χαμηλής φόρτισης για παρατεταμένο χρονικό διάστημα, ο θεικός μολύβδος κρυσταλλοποιείται επάνω στις πλάκες. Οι κρύσταλλοι αυτοί μειώνουν την ενεργή επιφάνεια του συσσωρευτή, αυξάνοντας την εσωτερική του αντίσταση (Internal Resistance - IR). Μια αυξημένη εσωτερική αντίσταση είναι το κυριότερο σύμπτωμα πτώσης του SoH [27].

Η συνεχής καταγραφή και ανάλυση αυτών των παραγόντων μέσω συστημάτων IoT, καθιστά εφικτή τη μετάβαση από τη διορθωτική συντήρηση (αντικατάσταση όταν χαλάσει) στην προγνωστική συντήρηση (Predictive Maintenance), αποφεύγοντας δαπανηρές διακοπές ρεύματος.

3.5 Βιβλιογραφική Ανασκόπηση Μεθόδων Εκτίμησης SoC και SoH

Οι Sreekanth, Anupalli και Sudheer [7], σε ανασκόπηση μεθόδων εκτίμησης SoC, SoH και SoP ειδικά για φωτοβολταϊκά συστήματα, επισημαίνουν ότι οι παραδοσιακές τεχνικές —όπως η Coulomb Counting και οι μέθοδοι βασισμένες σε μοντέλο— συχνά αδυνατούν να ανταποκριθούν επαρκώς στις δυναμικές συνθήκες λειτουργίας ενός Φ/Β συστήματος, καθώς είναι ευαίσθητες σε μεταβολές θερμοκρασίας, φαινόμενα γήρανσης και ασταθείς κύκλους φόρτισης-εκφόρτισης. Οι ίδιοι συγγραφείς

αναγνωρίζουν ότι, αν και οι σύγχρονες τεχνικές μηχανικής μάθησης και τα υβριδικά μοντέλα προσφέρουν βελτιωμένη ακρίβεια και προσαρμοστικότητα, εισάγουν παράλληλα σημαντικές απαιτήσεις σε δεδομένα εκπαίδευσης και υπολογιστικό φόρτο, κάτι ιδιαίτερα δύσκολο να ικανοποιηθεί σε εφαρμογές πραγματικού χρόνου με περιορισμένους πόρους [7]. Η παρατήρηση αυτή έρχεται σε άμεση αντιστοιχία με μια βασική σχεδιαστική επιλογή της παρούσας εργασίας: αντί για ένα μοντέλο μηχανικής μάθησης, υιοθετήθηκε μια προσέγγιση που χρησιμοποιεί την OCV για την αρχική εκτίμηση του SoC και τη μέθοδο Coulomb Counting για τη μετέπειτα δυναμική παρακολούθησή του, προσαρμοσμένη στους περιορισμένους υπολογιστικούς πόρους ενός μικροελεγκτή χαμηλού κόστους. Στο ίδιο πνεύμα κινείται και η εργασία των Loukil, Masmoudi και Derbel [26], η οποία αφορά ιδιαίτερα την παρούσα εργασία καθώς χρησιμοποιεί συσσωρευτή μολύβδου-οξέος 12V/7Ah —παρόμοιας κλίμακας με τον VRLA συσσωρευτή που δοκιμάστηκε στο Κεφάλαιο 10. Οι συγγραφείς ανέπτυξαν ένα ισοδύναμο ηλεκτρικό κύκλωμα δεύτερης τάξης (RC model), προσδιόρισαν πειραματικά τις παραμέτρους του και εφάρμοσαν φίλτρο Kalman διακριτού χρόνου για να εκτιμήσουν τις εσωτερικές τάσεις του μοντέλου και, μέσω αυτών, την τάση ανοιχτού κυκλώματος (OCV) και το SoC. Ιδιαίτερα χρήσιμη για την παρούσα εργασία είναι η ρητή διαπίστωση των συγγραφέων ότι, ενώ τόσο η Coulomb Counting όσο και το φίλτρο Kalman οδηγούν σε ακριβή εκτίμηση SoC, η πρώτη προϋποθέτει γνωστή αρχική τιμή SoC, περιορισμός που το φίλτρο Kalman ξεπερνά εγγενώς. Η παρούσα εργασία αντιμετωπίζει το ίδιο πρόβλημα με μια ηθελημένα απλούστερη υβριδική προσέγγιση: χρησιμοποιεί την OCV για την αρχικοποίηση της εκτίμησης του SoC και στη συνέχεια τη μέθοδο Coulomb Counting για τη δυναμική ενημέρωσή του, εξοικονομώντας υπολογιστικούς πόρους στον μικροελεγκτή.

Τέλος, οι Yu κ.ά. [27] παρουσιάζουν εκτενή ανασκόπηση των μηχανισμών γήρανσης VRLA συσσωρευτών σε εφαρμογές εφεδρικής τροφοδοσίας, υπογραμμίζοντας ότι η ανάγκη για ακριβή, σε πραγματικό χρόνο εκτίμηση της κατάστασης υγείας (SoH) παραμένει κρίσιμη, καθώς σοβαρά περιστατικά αστοχίας σε συστήματα ισχύος οφείλονται συχνά σε μη έγκαιρα διαγνωσμένη γήρανση συσσωρευτών VRLA. Η διαπίστωση αυτή ενισχύει περαιτέρω το θεωρητικό υπόβαθρο των μηχανισμών προστασίας και καταγραφής φθοράς που υιοθετήθηκαν στην παρούσα μελέτη.

Κεφάλαιο 4ο: Συνολική Αρχιτεκτονική του Συστήματος

4.1 Εισαγωγή

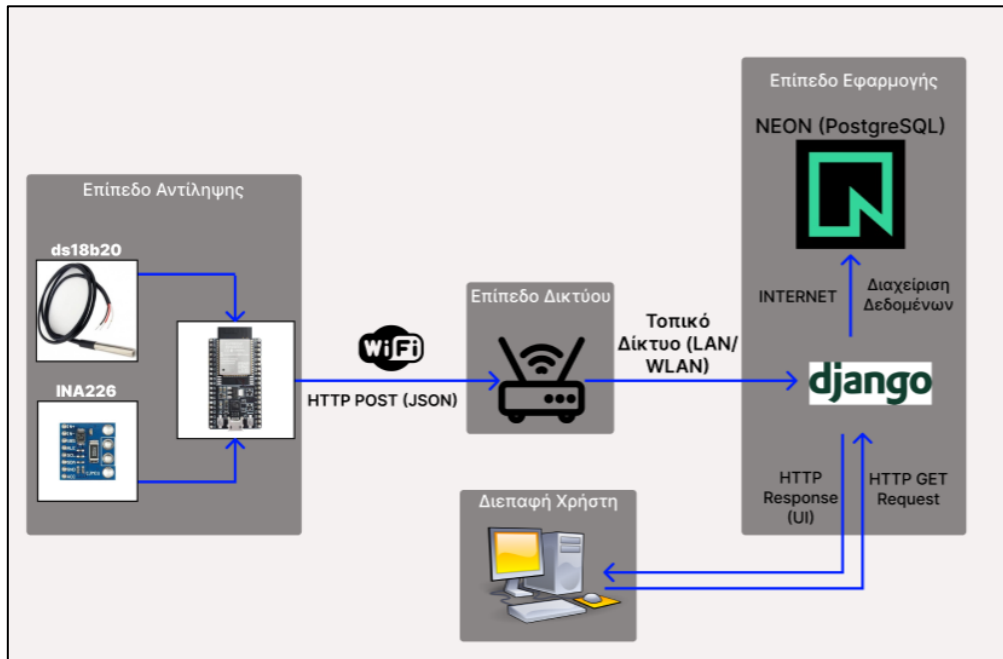
Στο παρόν κεφάλαιο παρουσιάζεται η συνολική αρχιτεκτονική του συστήματος Διαδικτύου των Πραγμάτων (IoT). Σκοπός του συστήματος είναι η αδιάλειπτη παρακολούθηση, καταγραφή και διαχείριση της κατάστασης φόρτισης (SoC) και της υγείας (SoH) μπαταριών σε αυτόνομες φωτοβολταϊκές εγκαταστάσεις. Το κεφάλαιο αναλύει τα δομικά χαρακτηριστικά της κατασκευής και περιγράφει τη ροή της πληροφορίας από το φυσικό επίπεδο των αισθητήρων (Hardware) έως το επίπεδο της διαδικτυακής εφαρμογής (Cloud/Backend), προσφέροντας μια ολοκληρωμένη εικόνα της διασύνδεσης και της αλληλεπίδρασης των επιμέρους υποσυστημάτων.

4.2 Χαρακτηριστικά του συστήματος

Τα βασικά χαρακτηριστικά του συστήματος είναι:

- **Λειτουργία σε Πραγματικό Χρόνο (Real-Time Monitoring):** Το σύστημα συλλέγει και μεταδίδει δεδομένα τάσης, ρεύματος και θερμοκρασίας σε συνεχή ροή, εξασφαλίζοντας την άμεση ενημέρωση του τελικού χρήστη.
- **Υπολογιστική στα Ακρα (Edge Computing & Local BMS):** Ο μικροελεγκτής δεν λειτουργεί ως απλός πομπός δεδομένων, αλλά ενσωματώνει λογική Τοπικού Συστήματος Διαχείρισης Μπαταρίας (Edge BMS). Δύναται να λαμβάνει άμεσες αποφάσεις αποκοπής ή επανασύνδεσης του φορτίου, μέσω του κυκλώματος MOSFET, για την προστασία της μπαταρίας από βαθιά εκφόρτιση ή υπερθέρμανση, χωρίς να εξαρτάται από την απόκριση του κεντρικού διαδικτυακού διακομιστή.
- **Υψηλή Ακρίβεια Μετρήσεων:** Μέσω της τεχνικής της υπερδειγματοληψίας (oversampling) στις αναγνώσεις των αισθητήρων, επιτυγχάνεται η εξομάλυνση του ηλεκτρικού θορύβου και η εξαγωγή εξαιρετικά αξιόπιστων τηλεμετρικών δεδομένων.
- **Ανοχή σε Σφάλματα (Fault Tolerance):** Ενσωματώνει μηχανισμούς αυτόματης επανασύνδεσης σε περίπτωση απώλειας του δικτύου Wi-Fi, δυναμικές ρουτίνες καθαρισμού μνήμης (Garbage Collection) για την αποφυγή υπερχειλίσεων, καθώς και διαχείριση σφαλμάτων μέσω εντολών επανεκκίνησης (Hardware Reset), εξασφαλίζοντας αδιάλειπτη λειτουργία χωρίς την ανάγκη ανθρώπινης παρέμβασης.
- **Ασφάλεια Δεδομένων και Κλιμακωσιμότητα:** Η επικοινωνία βασίζεται σε ασφαλή Web APIs, ενώ η κεντρική διαχείριση των δεδομένων πραγματοποιείται μέσω ενός σύγχρονου συστήματος διαχείρισης σχεσιακών βάσεων δεδομένων (PostgreSQL), επιτρέποντας την ταυτόχρονη παρακολούθηση πολλαπλών κόμβων-μπαταριών.

4.3 Γενική περιγραφή της εφαρμογής



Σχήμα 4.1: Αρχιτεκτονική του Συστήματος

Μια γενική περιγραφή της εφαρμογής με βάση το Σχήμα 4.1 είναι η παρακάτω:

Αρχικά, ο μικροελεγκτής ESP32 αναγνωρίζει και διαβάζει τις τιμές της τάσης και του ρεύματος από τον αισθητήρα INA226, καθώς και τη θερμοκρασία της μπαταρίας από τον DS18B20. Ταυτόχρονα, λειτουργώντας ως τοπικό σύστημα ελέγχου, ενεργοποιεί ή απενεργοποιεί το φορτίο μέσω ενός κυκλώματος MOSFET για την προστασία της μπαταρίας από ακραίες καταστάσεις. Στη συνέχεια, αφού συνδεθεί στο δίκτυο Wi-Fi που έχει αποθηκευμένο στη μνήμη του, ομαδοποιεί τις μετρήσεις σε μορφή JSON και τις αποστέλλει μέσω του πρωτοκόλλου HTTP POST στον κεντρικό τοπικό διακομιστή. Εκεί, μέσω της αρχιτεκτονικής του Django, τα δεδομένα παραλαμβάνονται από το API και αποθηκεύονται σε μία σχεσιακή βάση δεδομένων PostgreSQL που φιλοξενείται στο υπολογιστικό νέφος (Neon). Τέλος, το ιστορικό των μετρήσεων από τη βάση δεδομένων είναι άμεσα διαθέσιμο στον χρήστη μέσω μιας δυναμικής ιστοσελίδας. Η διεπαφή αυτή αξιοποιεί τη βιβλιοθήκη Chart.js για να δημιουργήσει διαδραστικά διαγράμματα, τα οποία αποτυπώνουν σε πραγματικό χρόνο την κατάσταση φόρτισης και την υγεία της εγκατάστασης.

4.4 Λειτουργικές και Μη Λειτουργικές Απαιτήσεις του Συστήματος

Πριν από τον σχεδιασμό της αρχιτεκτονικής, κρίθηκε απαραίτητο να οριστούν με σαφήνεια οι απαιτήσεις που καλείται να ικανοποιήσει το σύστημα, με βάση τα προβλήματα που εντοπίστηκαν στην ενότητα 1.2. Οι απαιτήσεις αυτές διακρίνονται σε λειτουργικές (functional), οι οποίες περιγράφουν τι πρέπει να κάνει το σύστημα, και σε μη λειτουργικές (non-functional), οι οποίες θέτουν ποιοτικά κριτήρια αποδοσης, αξιοπιστίας και ασφάλειας που πρέπει να διέπουν τη λειτουργία του.

4.4.1 Λειτουργικές Απαιτήσεις

Ο Πίνακας 4.1 συνοψίζει τις βασικές λειτουργικές απαιτήσεις του συστήματος.

Πίνακας 4.1: Λειτουργικές Απαιτήσεις

Κωδικός	Περιγραφή Απαίτησης
FR-01	Το σύστημα συλλέγει σε πραγματικό χρόνο μετρήσεις τάσης, ρεύματος και θερμοκρασίας του συσσωρευτή μέσω των αισθητήρων INA226 και DS18B20.
FR-02	Ο μικροελεγκτής αποφασίζει τοπικά (Edge) την αποκοπή ή επανασύνδεση του φορτίου μέσω του κυκλώματος διπλού MOSFET, χωρίς να απαιτείται απόκριση από τον απομακρυσμένο διακομιστή.
FR-03	Τα τηλεμετρικά δεδομένα μορφοποιούνται σε πακέτα JSON και αποστέλλονται περιοδικά στο backend μέσω του πρωτοκόλλου HTTP POST.
FR-04	Το backend υπολογίζει τους δείκτες SoC, SoH, Βάθους Εκφόρτισης (DoD) και Εσωτερικής Αντίστασης (IR) με βάση τα εισερχόμενα δεδομένα και το ιστορικό μετρήσεων.
FR-05	Το σύστημα υποστηρίζει ταυτόχρονη παρακολούθηση πολλαπλών συσκευών-κόμβων (fleet) από έναν ενιαίο λογαριασμό διαχειριστή.
FR-06	Το σύστημα αποστέλλει αυτοματοποιημένη ειδοποίηση μέσω ηλεκτρονικού ταχυδρομείου όταν εντοπίζονται κρίσιμες συνθήκες λειτουργίας.
FR-07	Ο χρήστης παρακολουθεί ιστορικά και τρέχοντα δεδομένα μέσω διαδραστικών γραφημάτων (Chart.js).

4.4.2 Μη Λειτουργικές Απαιτήσεις

Πέραν των λειτουργικών απαιτήσεων, το σύστημα σχεδιάστηκε ώστε να ικανοποιεί και μια σειρά ποιοτικών κριτηρίων, τα οποία συνοψίζονται στον Πίνακα 4.2.

Πίνακας 4.2: Μη Λειτουργικές Απαιτήσεις

Κωδικός	Κατηγορία	Περιγραφή Απαίτησης
NFR-01	Αξιοπιστία	Το σύστημα συνεχίζει τη λειτουργία του χωρίς ανθρώπινη παρέμβαση, ακόμη και σε περίπτωση προσωρινής απώλειας του Wi-Fi ή σφάλματος μνήμης.

NFR-02	Απόκριση σε Πραγματικό Χρόνο	Η ενεργοποίηση του μηχανισμού προστασίας (LVD) πραγματοποιείται σε χρόνο της τάξης του δευτερολέπτου, ανεξάρτητα από την κατάσταση του δικτύου.
NFR-03	Χαμηλό Κόστος	Το συνολικό κόστος υλικού παραμένει σημαντικά χαμηλότερο από αντίστοιχα εμπορικά συστήματα BMS κλειστής αρχιτεκτονικής.
NFR-04	Κλιμακωσιμότητα	Η προσθήκη νέων συσκευών-κόμβων στο σύστημα δεν απαιτεί μεταβολές στην υπάρχουσα υποδομή του backend.
NFR-05	Ασφάλεια	Η πρόσβαση στο API και στα δεδομένα προστατεύεται μέσω ταυτοποίησης συσκευής (MAC address) και ταυτοποίησης χρήστη (session-based authentication).
NFR-06	Ευχρηστία	Η διεπαφή είναι προσβάσιμη μέσω απλού φυλλομετρητή, χωρίς την ανάγκη εγκατάστασης εξειδικευμένου λογισμικού από τον τελικό χρήστη.
NFR-07	Συντηρησιμότητα	Η χρήση της αρχιτεκτονικής MTV και τεχνολογιών ανοιχτού κώδικα διευκολύνει τη μελλοντική επέκταση και συντήρηση του συστήματος.

Οι παραπάνω απαιτήσεις αποτέλεσαν τον βασικό οδηγό για τις σχεδιαστικές αποφάσεις που περιγράφονται στη συνέχεια του παρόντος κεφαλαίου, καθώς και για την επιλογή των επιμέρους τεχνολογιών που αναλύονται στα Κεφάλαια 5 έως 8.

4.5 Εναλλακτικές Αρχιτεκτονικές Επιλογές και Τεκμηρίωση Σχεδιαστικών Αποφάσεων

Κατά τη φάση σχεδιασμού του συστήματος εξετάστηκαν εναλλακτικές τεχνολογικές επιλογές για κάθε επίπεδο της αρχιτεκτονικής. Στην παρούσα ενότητα παρουσιάζονται συνοπτικά οι κυριότερες εναλλακτικές, μαζί με την αιτιολόγηση της τελικής επιλογής, με στόχο να καταστεί σαφές ότι η αρχιτεκτονική δεν προέκυψε τυχαία, αλλά ως αποτέλεσμα στάθμισης κόστους, απόδοσης και κάλυψης των απαιτήσεων της ενότητας 4.4.

4.5.1 Επιλογή Μικροελεγκτή

Για το Επίπεδο Αντίληψης εξετάστηκαν τρεις εναλλακτικές πλατφόρμες: ο μικροελεγκτής Arduino Uno/Nano, ο μικροϋπολογιστής Raspberry Pi και ο τελικά επιλεγείς ESP32. Το Arduino, παρότι οικονομικό και ευρέως τεκμηριωμένο, δεν διαθέτει ενσωματωμένη συνδεσιμότητα Wi-Fi, γεγονός που θα απαιτούσε την προσθήκη ξεχωριστής μονάδας δικτύου και επιπλέον πολυπλοκότητα στη διαχείριση επικοινωνίας. Το Raspberry Pi, από την άλλη πλευρά, προσφέρει πλήρες λειτουργικό σύστημα και μεγαλύτερη υπολογιστική ισχύ, κρίνεται όμως υπερβολικό (over-engineered) για τις ανάγκες ενός κόμβου Edge BMS, ενώ η υψηλότερη κατανάλωση ενέργειας και το μεγαλύτερο κόστος κτήσης έρχονται σε αντίθεση με την απαίτηση NFR-03. Ο ESP32 επιλέχθηκε επειδή συνδυάζει διπύρνηνο επεξεργαστή, ενσωματωμένο υποσύστημα Wi-Fi, χαμηλή κατανάλωση ενέργειας και εγγενή υποστήριξη πρωτοκόλλων όπως το I2C και το 1-Wire, σε τιμή κτήσης πολλαπλάσια χαμηλότερη από αυτή ενός Raspberry Pi [15]. Ο συνδυασμός αυτός τον καθιστά ιδανικό για την ταυτόχρονη εκτέλεση της λογικής του Edge BMS και της ασύρματης μετάδοσης δεδομένων, χωρίς να απαιτείται δεύτερη συσκευή.

4.5.2 Επιλογή Πρωτοκόλλου Επικοινωνίας

Σε επίπεδο συστήματος, εξετάστηκε επίσης η χρήση του πρωτοκόλλου MQTT ως εναλλακτική του HTTP POST για τη μεταφορά των τηλεμετρικών δεδομένων. Η αναλυτική σύγκριση των δύο πρωτοκόλλων, καθώς και τα κριτήρια που οδήγησαν στην τελική επιλογή, παρουσιάζονται διεξοδικά στην ενότητα 7.3.1. Σε επίπεδο αρχιτεκτονικής αξίζει να σημειωθεί ότι η επιλογή του HTTP POST εξυπηρέτησε την απλότητα υλοποίησης και την άμεση συμβατότητα με το Django REST API, χωρίς να απαιτείται η ανάπτυξη ξεχωριστού message broker, γεγονός που ταιριάζει με την κλίμακα του παρόντος συστήματος.

4.5.3 Επιλογή Υποδομής Αποθήκευσης Δεδομένων

Για το Επίπεδο Εφαρμογής εξετάστηκε η χρήση μιας τοπικά φιλοξενούμενης βάσης δεδομένων (π.χ. SQLite ή τοπική εγκατάσταση PostgreSQL) έναντι μιας serverless λύσης στο υπολογιστικό νέφος. Η τοπική βάση θα απλοποιούσε την αρχική εγκατάσταση, θα περιόριζε όμως σημαντικά τη δυνατότητα απομακρυσμένης πρόσβασης πολλαπλών χρηστών και θα απαιτούσε πρόσθετη μέριμνα για αντίγραφα ασφαλείας, διαθεσιμότητα και συντήρηση του διακομιστή βάσης δεδομένων — ευθύνες που ξεπερνούν το πλαίσιο μιας διπλωματικής εργασίας. Η επιλογή της Neon, μιας serverless πλατφόρμας φιλοξενίας PostgreSQL, κάλυψε τις απαιτήσεις NFR-04 (κλιμακωσιμότητα) και NFR-07 (συντηρησιμότητα), καθώς προσφέρει αυτόματη διαχείριση πόρων, δυνατότητα πρόσβασης από οπουδήποτε και μηδενική επιβάρυνση σε διαχείριση υποδομής, στοιχεία ιδιαίτερα χρήσιμα για τη λειτουργία του υποσυστήματος Fleet Management [22].

4.5.4 Κατανομή Λειτουργικότητας μεταξύ Edge και Cloud

Μια από τις κεντρικές σχεδιαστικές αποφάσεις της αρχιτεκτονικής αφορά την κατανομή της λογικής επεξεργασίας μεταξύ του μικροελεγκτή (Edge) και του διακομιστή (Cloud). Θα ήταν εφικτό η απόφαση αποκοπής του φορτίου να λαμβάνεται αποκλειστικά από το backend, μετά από επεξεργασία των εισερχόμενων δεδομένων. Η προσέγγιση αυτή απορρίφθηκε, καθώς θα καθιστούσε την προστασία της μπαταρίας πλήρως εξαρτημένη από τη διαθεσιμότητα του δικτύου Wi-Fi και του απομακρυσμένου διακομιστή, παραβιάζοντας την απαίτηση NFR-02. Αντίθετα, οι κρίσιμες αποφάσεις προστασίας (LVD) υλοποιήθηκαν τοπικά στο επίπεδο του μικροελεγκτή, ώστε να διασφαλίζεται άμεση απόκριση ακόμη

και σε συνθήκες απώλειας δικτύου, ενώ οι υπολογιστικά βαρύτεροι αλγόριθμοι εκτίμησης SoC/SoH — οι οποίοι απαιτούν πρόσβαση σε ιστορικά δεδομένα και μεγαλύτερη υπολογιστική ισχύ — ανατέθηκαν στο backend. Η κατανομή αυτή αντικατοπτρίζει τη φιλοσοφία της Υπολογιστικής στα Άκρα (Edge Computing), όπως περιγράφηκε συνοπτικά στην ενότητα 4.2, και αποτελεί βασικό στοιχείο διαφοροποίησης του συστήματος από απλές λύσεις τηλεμετρίας που βασίζονται αποκλειστικά σε κεντρική επεξεργασία.

4.5.5 Σύγκριση με Συναφή Συστήματα IoT-BMS της Βιβλιογραφίας

Προκειμένου να τεκμηριωθούν πληρέστερα οι σχεδιαστικές αποφάσεις που αναλύθηκαν στις προηγούμενες υποενότητες (επιλογή μονού μικροελεγκτή ESP32, τοπική αποκοπή φορτίου Edge BMS, ανοιχτού κώδικα backend), κρίνεται σκόπιμη η σύγκριση της προτεινόμενης αρχιτεκτονικής με συναφείς υλοποιήσεις της διεθνούς βιβλιογραφίας.

Πιο κοντά στο αντικείμενο της παρούσας εργασίας βρίσκεται η δημοσίευση των Vyshnav κ.ά. [6], η οποία παρουσιάζει ένα IoT-ενσωματωμένο σύστημα διαχείρισης μπαταρίας που υποστηρίζει συσσωρευτές λιθίου-ιόντων και μολύβδου-οξέος. Το σύστημα βασίζεται σε μικροελεγκτή ESP32 για τη συλλογή μετρήσεων τάσης, ρεύματος (μέσω INA219) και θερμοκρασίας, τον υπολογισμό των δεικτών SOC, SOH, SOP και τη μετάδοσή τους μέσω Wi-Fi προς την πλατφόρμα Firebase, από όπου γίνονται προσβάσιμα μέσω εφαρμογής κινητού (Flutter) και τοπικής οθόνης LCD. Ιδιαίτερο χαρακτηριστικό του συστήματος είναι η χρήση δευτέρου μικροελεγκτή (Arduino Nano) αποκλειστικά για μηχανισμό πυρασφάλειας με απελευθέρωση CO₂.

Παρά τη θεματική εγγύτητα, η σύγκριση των αρχιτεκτονικών αναδεικνύει σαφείς διαφοροποιήσεις, οι οποίες συνοψίζονται στον Πίνακα 4.3:

Πίνακας 4.3: Σύγκριση με το Σύστημα των Vyshnav κ.ά. [6]

Χαρακτηριστικό	Vyshnav κ.ά. [6]	Παρούσα Εργασία
Χημεία μπαταρίας	Λιθίου-ιόντων & μολύβδου-οξέος	Μολύβδου-οξέος (VRLA)
Μικροελεγκτές	ESP32 + ξεχωριστός Arduino Nano	Ένας ESP32
Αυτόματη αποκοπή φορτίου (LVD)	Δεν προκύπτει από την περιγραφή	Ναι, τοπικά στο edge
Μέθοδος SoC	Coulomb Counting	OCV + Coulomb Counting
Backend	Firebase (διαχειριζόμενο)	Django + PostgreSQL/Neon (Ανοιχτού Κώδικα Backend / Serverless Neon)
Διαχείριση πολλαπλών συσκευών	Δεν αναφέρεται	Ναι (Fleet Management)
Μηχανισμός πυρασφάλειας	Ναι (αισθητήρας φλόγας + CO ₂)	Όχι (μη σχετικό ρίσκο για VRLA)

Η πιο ουσιαστική διαφορά αφορά τον μηχανισμό προστασίας: το σύστημα των Vyshnav κ.ά. ενημερώνει τον χρήστη μέσω οθόνης και εφαρμογής όταν εντοπιστεί πρόβλημα, χωρίς αυτόματη τοπική

αποκοπή του φορτίου σε περίπτωση βαθιάς εκφόρτισης. Αντίθετα, ο πυρήνας της αρχιτεκτονικής της παρούσας εργασίας είναι η αυτόνομη τοπική παρέμβαση (LVD) απευθείας στο edge, ανεξάρτητα από τη διαθεσιμότητα δικτύου. Δεύτερον, η μέθοδος Coulomb Counting του [6] δεν συνοδεύεται από επαναβαθμονόμηση OCV, παραμένοντας ευάλωτη σε συσσωρευμένο σφάλμα. Τρίτον, η χρήση δύο ξεχωριστών μικροελεγκτών αυξάνει την πολυπλοκότητα συγκριτικά με την ενιαία υλοποίηση σε έναν μόνο ESP32. Τέλος, η προτεινόμενη λύση παρέχει δυνατότητα κεντρικής διαχείρισης στόλου (Fleet Management) και βασίζεται σε πλήρως αυτοφιλοξενούμενη υποδομή ανοιχτού κώδικα (Django/PostgreSQL) αντί για ιδιοταγείς, διαχειριζόμενες υπηρεσίες όπως το Firebase.

Επιπλέον, αξίζει να σημειωθεί η διαφοροποίηση ως προς τις υπολογιζόμενες μετρικές: ενώ το σύστημα των Vyshnav κ.ά. [6] επεκτείνεται στον υπολογισμό και της Κατάστασης Ισχύος (State of Power - SoP), στην παρούσα εργασία κρίθηκε σκόπιμη η παράλειψή της. Ο δείκτης SoP εκφράζει τη μέγιστη ισχύ (σε W) που δύναται να αποδώσει ή να απορροφήσει στιγμιαία ο συσσωρευτής χωρίς να παραβιαστούν τα ασφαλή όρια τάσης, ρεύματος και θερμοκρασίας. Ο δείκτης αυτός είναι πρωταρχικής σημασίας σε εφαρμογές ηλεκτροκίνησης (Electric Vehicles), όπου παρατηρούνται βίαιες επιταχύνσεις ή ανάκτηση ενέργειας κατά την πέδηση. Αντίθετα, στο πλαίσιο ενός μικρού, αυτόνομου φωτοβολταϊκού συστήματος σταθερής εγκατάστασης, τα φορτία είναι κατά βάση στατικά ή προβλέψιμα (π.χ. συνεχής φωτισμός DC ή λειτουργία χαμηλής κατανάλωσης). Συνεπώς, η συνεχής καταγραφή της πραγματικής στιγμιαίας ισχύος σε πραγματικό χρόνο, σε συνδυασμό με την αυτόνομη προστασία αποκοπής φορτίου (LVD), καλύπτει πλήρως τις λειτουργικές απαιτήσεις ασφαλείας, απαλλάσσοντας τον μικροελεγκτή από περιττό υπολογιστικό φόρτο.

Κεφάλαιο 5ο: Εργαλεία Ανάπτυξης και Γλώσσες Προγραμματισμού

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλύσουμε τις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος, frameworks καθώς και τα περιβάλλοντα για την ανάπτυξη και την συγγραφή του προγράμματος.

Για τον προγραμματισμό του ESP32 επιλέχθηκε η MicroPython. Πρόκειται για μια εξαιρετικά βελτιστοποιημένη έκδοχή της Python 3, σχεδιασμένη ειδικά για μικροελεγκτές και περιβάλλοντα με αυστηρά περιορισμένους πόρους μνήμης.

Το Περιβάλλον Ολοκληρωμένης Ανάπτυξης (IDE) που επιλέχθηκε για την συγγραφή των προγραμμάτων Python, HTML, CSS και JavaScript είναι Visual Studio Code (VS Code) της Microsoft. Για τον προγραμματισμό του μικροελεγκτή ESP32 χρησιμοποιήθηκε το Thonny IDE. Τέλος, όλα τα αρχεία αποθηκεύονται στο GitHub.

5.2 MicroPython

Σε αντίθεση με την παραδοσιακή προσέγγιση της γλώσσας C/C++ για τον προγραμματισμό του μικροελεγκτή ESP32 επιλέχθηκε η MicroPython. Η συγκεκριμένη γλώσσα είναι ειδικά σχεδιασμένη για μικροελεγκτές και περιβάλλοντα με αυστηρά περιορισμένους πόρους μνήμης. Προσφέρει πολλά πλεονεκτήματα στην ταχύτητα ανάπτυξης του προγράμματος και στην αποστολή των δεδομένων προς το WEB API μέσω JSON. Στον Πίνακα 5.1 υπάρχει ένα στιγμιότυπο κώδικα από το αρχείο main.py όπου ο μικροελεγκτής ελέγχει την τάση και τη θερμοκρασία της μπαταρίας και αποφασίζει αν θα αποσυνδέσει ή θα επανασυνδέσει το φορτίο, ενεργοποιώντας ή απενεργοποιώντας μια ακίδα εξόδου (mosfet.value()):

Πίνακας 5.1: Παράδειγμα κώδικα MicroPython

```
if real_voltage < 11.5 or temp_c > 45.0:
    mosfet.value(0)

elif real_voltage > 12.5 and temp_c < 40.0:
    mosfet.value(1)
```

5.3 Python

Για την συγγραφή των προγραμμάτων (Backend) χρησιμοποιήθηκε η Python, μία από τις πιο δημοφιλείς και ισχυρές γλώσσες προγραμματισμού στον κόσμο, και συγκεκριμένα η έκδοση 3.12. Η Python είναι μια διερμηνευόμενη (interpreted) γλώσσα. Δηλαδή, αντίθετα με γλώσσες όπως η C++ ή η Java οι οποίες πρώτα μεταφράζονται σε γλώσσα μηχανής (compile), η Python χρησιμοποιεί ένα πρόγραμμα που λέγεται Ερμηνευτής (Interpreter). Αυτός διαβάζει και εκτελεί τον κώδικα γραμμή προς γραμμή, τη στιγμή ακριβώς που του το ζητάει ο χρήστης.

Για την παρούσα εργασία χρησιμοποιήθηκαν τα παρακάτω αρχεία Python (.py):

Στον διακομιστή:

- **models.py:** Εδώ γράφτηκαν οι κλάσεις της βάσης δεδομένων (π.χ. Device και Measurement). Είναι ο ORM κώδικας που "μιλάει" στην PostgreSQL.
- **views.py:** Εδώ βρίσκεται όλη η λογική του συστήματος.
- **urls.py:** Το αρχείο που αναλαμβάνει τη "δρομολόγηση". Αντιστοιχεί τα URLs (π.χ. /api/sensor-data/) με τις σωστές συναρτήσεις του views.py.
- **admin.py:** Εδώ γράφτηκε ο κώδικας για να εμφανίζονται όμορφα οι μπαταρίες στο έτοιμο περιβάλλον διαχείρισης του Django.
- **settings.py:** Το αρχείο ρυθμίσεων, όπου ρυθμίζονται η PostgreSQL, τα emails, η ασφάλεια κ.λπ.

Στον μικροελεγκτή:

- **main.py:** Το αρχείο που τρέχει ασταμάτητα μέσα στον μικροελεγκτή. Διαβάζει τους αισθητήρες (π.χ. θερμοκρασία μέσω DS18B20), φτιάχνει το JSON και το στέλνει μέσω Wi-Fi.
- Παρακάτω στον πίνακα 5.2 δίνεται ένα παράδειγμα σε κώδικα Python, και συγκεκριμένα της ανάγνωσης αισθητήρα INA226 με oversampling του αρχείου main.py. Τα υπόλοιπα αρχεία βρίσκονται στο ΠΑΡΑΡΤΗΜΑ Α.

Πίνακας 5.2: Παράδειγμα κώδικα Python

```
def read_ina_register(reg):
    data = i2c.readfrom_mem(INA226_ADDR, reg, 2)
    return ustruct.unpack('>h', data)[0]

def get_averaged_ina_readings(samples=5):
    sum_v, sum_i = 0.0, 0.0
    for _ in range(samples):
        raw_bus = read_ina_register(0x02)
        sum_v += raw_bus * 0.00125
        raw_shunt = read_ina_register(0x01)
        sum_i += (raw_shunt * 0.0000025) / R_TOTAL
        time.sleep_ms(50)
    return round(sum_v / samples, 2), round(sum_i / samples, 3)
```

Όπως αποτυπώνεται στον Πίνακα 5.2, η διαδικασία λήψης των ηλεκτρικών μετρήσεων από τον αισθητήρα INA226 χωρίζεται σε δύο συναρτήσεις:

1. Η συνάρτηση `read_ina_register(reg)`: Αναλαμβάνει τη χαμηλού επιπέδου επικοινωνία I2C με τον αισθητήρα. Διαβάζει 2 διαδοχικά bytes από τη διεύθυνση μνήμης του επιθυμητού καταχωρητή (reg) στη διεύθυνση INA226_ADDR. Στη συνέχεια, μέσω της μεθόδου `ustruct.unpack('>h', data)[0]`, μετατρέπει τα ακατέργαστα δεδομένα από μορφή big-endian σε προσημασμένο ακέραιο 16-bit (signed short), αποδίδοντας την αριθμητική τιμή της μέτρησης.
2. Η συνάρτηση `get_averaged_ina_readings(samples=5)`: Υλοποιεί την τεχνική της υπερδειγματοληψίας (oversampling). Μέσα από έναν βρόχο 5 επαναλήψεων με ενδιάμεση παύση 50ms, εκτελεί διαδοχικές αναγνώσεις:
 - Διαβάζει τον καταχωρητή τάσης διαύλου 0x02 (Bus Voltage Register) και πολλαπλασιάζει την τιμή με το βήμα κβάντισης 0.00125V (1.25 mV/LSB), όπως

ορίζεται από το datasheet του ολοκληρωμένου, μετατρέποντας το μέγεθος απευθείας σε Volts.

- Διαβάζει τον καταχωρητή πτώσης τάσης διακλάδωσης 0x01 (Shunt Voltage Register), πολλαπλασιάζει με τον συντελεστή 0.0000025V (2.5 μ V/LSB), και διαιρεί με τη συνολική ισοδύναμη αντίσταση διακλάδωσης R_TOTAL. Βάσει του νόμου του Ohm, εξάγεται το ρεύμα σε Amperes.
- Τέλος, επιστρέφει τον μέσο όρο των 5 δειγμάτων στρογγυλοποιημένο, εξαλείφοντας αποτελεσματικά τον ηλεκτρικό θόρυβο υψηλής συχνότητας που προκαλείται από τις μεταγωγές του ρυθμιστή PWM.

5.4 Τεχνολογίες Ιστού (HTML, JavaScript, Chart.js)

Η HTML σε συνδυασμό με την γλώσσα JavaScript συμβάλλουν στον σχεδιασμό της ιστοσελίδας, η οποία θα χρησιμοποιηθεί για την παρακολούθηση των διαγραμμάτων. Ακόμη θα χρησιμοποιηθεί η βιβλιοθήκη οπτικοποίησης της JavaScript, Chart.js για τη δημιουργία αυτών των διαγραμμάτων και γραφημάτων της ιστοσελίδας.

Στο στιγμιότυπο που παρουσιάζεται στον Πίνακα 5.3 από το αρχείο home.html, παρουσιάζεται η γέφυρα επικοινωνίας για τη δημιουργία του γραφήματος της Τάσης (Voltage):

Πίνακας 5.3: Παράδειγμα κώδικα HTML – JavaScript – Chart.js

```
<div class="card dashboard-card">
  <canvas id="voltageChart"></canvas>
</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
  const labels = JSON.parse('{{ timestamps|safe }}');
  const voltageData = JSON.parse('{{ voltage_data|safe }}');
  new Chart(document.getElementById('voltageChart').getContext('2d'), {
    type: 'line',
    data: {
      labels: labels, // Ο άξονας X (Χρονικές στιγμές)
      datasets: [{
        label: 'Τάση (Voltage V)',
        data: voltageData, // Ο άξονας Y (Τιμές τάσης)
        borderColor: '#3273F6',
        backgroundColor: 'rgba(50, 115, 246, 0.2)',
        borderWidth: 2,
        fill: true,
```

```

        tension: 0.4 // Δημιουργεί καμπυλότητα στη γραμμή
    }]
}
});
</script>

```

Για την υλοποίηση των δυναμικών γραφημάτων στη διεπαφή χρήστη, απαιτήθηκε η στενή συνεργασία της HTML με τη JavaScript. Αρχικά, δεσμεύεται ο απαραίτητος χώρος στο template μέσω του στοιχείου `<canvas>`. Στη συνέχεια, τα δεδομένα που υπολογίζονται στο παρασκήνιο από το Django μορφοποιούνται σε JSON και μεταφέρονται με ασφάλεια στο περιβάλλον της JavaScript για να τροφοδοτήσουν τη βιβλιοθήκη Chart.js.

5.5 PostgreSQL

Για την αποθήκευση και ορθή διαχείριση των δεδομένων της παρούσας εργασίας, επιλέχθηκε η PostgreSQL [30]. Πρόκειται για ένα από τα πιο προηγμένα, ανοιχτού κώδικα Συστήματα Διαχείρισης Αντικειμενοστρεφών-Σχεσιακών Βάσεων Δεδομένων (ORDBMS). Η επιλογή της συγκεκριμένης τεχνολογίας βασίστηκε στην υψηλή αξιοπιστία της και την αυστηρή τήρηση των προτύπων ACID.

5.6 Django Framework

Το Django, ένα από τα δημοφιλέστερα και ισχυρότερα Web Frameworks, επιλέχθηκε για την ανάπτυξη της διαδικτυακής πλατφόρμας σε Python. Στην παρούσα εργασία, το Django ανέλαβε τη διαχείριση των χρηστών, τη δημιουργία του API και την επικοινωνία με τη βάση δεδομένων μέσω του δικού του ORM (Object-Relational Mapper).

5.7 Thonny IDE

Το Thonny χρησιμοποιήθηκε αποκλειστικά για τον προγραμματισμό του μικροελεγκτή ESP32. Είναι το κατεξοχήν προτεινόμενο περιβάλλον για την ανάπτυξη εφαρμογών σε MicroPython, καθώς έχει σχεδιαστεί με γνώμονα την απλότητα. Προσφέρει ενσωματωμένα εργαλεία για την εγκατάσταση του απαραίτητου υλικολογισμικού (firmware flashing) στην πλακέτα, άμεση αλληλεπίδραση με τον μικροελεγκτή μέσω της γραμμής εντολών REPL (Read-Evaluate-Print Loop), καθώς και εύκολη διαχείριση και μεταφορά των αρχείων κώδικα στο εσωτερικό σύστημα αρχείων του ESP32.

5.8 Visual Studio Code

Ως κύριο Περιβάλλον Ολοκληρωμένης Ανάπτυξης (IDE) επιλέχθηκε το Visual Studio Code. Πρόκειται για έναν ανοιχτού κώδικα, δωρεάν και εξαιρετικά επεκτάσιμο επεξεργαστή κώδικα, ο οποίος υποστηρίζει πολλαπλές γλώσσες όπως C++, TypeScript, PHP κ.ά. και υποστηρίζεται σε τρία λειτουργικά συστήματα, Windows, Linux και macOS. Το Visual Studio Code διαθέτει μεγάλο αριθμό επεκτάσεων (extensions), οι οποίες ενισχύουν τη διαδικασία συγγραφής κώδικα και τη συνολική λειτουργικότητα του περιβάλλοντος. Επιπλέον, λειτουργίες όπως η αυτόματη συμπλήρωση κώδικα (auto-completion) και η αποτελεσματική διαχείριση αρχείων συμβάλλουν στην ευελιξία και την ευχρηστία του. Η ενσωμάτωση τερματικού (terminal) εντός του περιβάλλοντος διευκόλυνε την ταυτόχρονη συγγραφή κώδικα και την εκτέλεση εντολών του διακομιστή Django. Το Visual Studio

Code είναι δωρεάν προς λήψη και χρήση και βρίσκεται στο παρακάτω link: <https://code.visualstudio.com/Download>.

5.9 GitHub

Το GitHub είναι διαδικτυακή πλατφόρμα φιλοξενίας κώδικα (hosting service) που βασίζεται στο κατανεμημένο σύστημα ελέγχου εκδόσεων Git. Είναι ένα βασικό εργαλείο για τους προγραμματιστές, καθώς επιτρέπει στους προγραμματιστές να παρακολουθούν κάθε αλλαγή κώδικα (commits), καθώς και να επιστρέφουν σε προηγούμενες εκδόσεις αν χρειαστεί. Ο πυρήνας αυτής της οργάνωσης είναι το Αποθετήριο (Repository), ένας δομημένος ψηφιακός χώρος που φιλοξενεί όχι μόνο τον πηγαίο κώδικα και την τεκμηρίωση (documentation) του έργου, αλλά διατηρεί και το πλήρες ιστορικό όλων των τροποποιήσεων. Επίσης, προσφέρει ένα δομημένο περιβάλλον για την ταυτόχρονη και απομακρυσμένη εργασία πολλών προγραμματιστών στο ίδιο έργο. Με λίγα λόγια, η διαδικασία ξεκινά με την προώθηση των τοπικών αλλαγών (push) που έχει κάνει ένα μέλος από ένα απομακρυσμένο αποθετήριο και στην συνέχεια μέσω των Pull Request μπορούν να προτείνουν αλλαγές, οι οποίες υπόκεινται σε αξιολόγηση πριν ενσωματωθούν οριακά στον κώδικα. Τέλος, η πλατφόρμα ενσωματώνει συστήματα Διαχείρισης Προβλημάτων (Issue Tracking), τα οποία επιτρέπουν την οργανωμένη καταγραφή σφαλμάτων (bugs) και νέων απαιτήσεων.

5.10 Υποδομές Εκτέλεσης και Φιλοξενίας

Για την εκτέλεση και τη λειτουργία των επιμέρους υποσυστημάτων της εφαρμογής, υιοθετήθηκε μια υβριδική αρχιτεκτονική η οποία συνδυάζει την τοπική υπολογιστική επεξεργασία με σύγχρονες υπηρεσίες υπολογιστικού νέφους (cloud). Πιο συγκεκριμένα, η εκτέλεση του κεντρικού διαδικτυακού διακομιστή (Django Backend) πραγματοποιείται σε τοπικό περιβάλλον (on-premises), εξασφαλίζοντας άμεσο έλεγχο του συστήματος, ελαχιστοποίηση των καθυστερήσεων (latency) κατά την επικοινωνία με τον μικροελεγκτή εντός του τοπικού δικτύου και ευελιξία κατά τη φάση των δοκιμών. Αντίθετα, για τη μόνιμη και ασφαλή αποθήκευση του μεγάλου όγκου των τηλεμετρικών δεδομένων, επιλέχθηκε η πλατφόρμα Neon [22] για τη φιλοξενία της σχεσιακής βάσης δεδομένων PostgreSQL. Η Neon αποτελεί μια σύγχρονη υποδομή τύπου serverless, η οποία διαχωρίζει τον υπολογιστικό φόρτο (compute) από την αποθήκευση δεδομένων (storage), διασφαλίζοντας κατ' αυτόν τον τρόπο υψηλή διαθεσιμότητα, ενισχυμένη ασφάλεια, καθώς και ταχύτατη απόκριση στα ερωτήματα που υποβάλλει και αναλύει το σύστημα IoT.

Κεφάλαιο 6ο: Επίπεδο Αντίληψης (Perception Layer)

6.1 Εισαγωγή

Σε μια αρχιτεκτονική Διαδικτύου των Πραγμάτων (IoT), το Επίπεδο Αντίληψης (Perception Layer) αποτελεί τη διεπαφή του συστήματος με τον φυσικό κόσμο. Στην παρούσα εργασία, αυτό το επίπεδο απαρτίζεται από τον μικροελεγκτή που αναλαμβάνει τον ρόλο της Κεντρικής Μονάδας Επεξεργασίας (CPU) του ενσωματωμένου συστήματος, καθώς και από τα αισθητήρια όργανα που καταγράφουν τη φυσική κατάσταση της μπαταρίας.

6.2 Ο Μικροελεγκτής ESP32 NodeMCU-32S

Ως "εγκέφαλος" του συστήματος Διαχείρισης Μπαταρίας (BMS) επιλέχθηκε η πλακέτα ανάπτυξης NodeMCU-32S, η οποία βασίζεται στο πανίσχυρο SoC (System on a Chip) ESP32 αναπτυγμένος από την Espressif Systems. Σε αντίθεση με τους παραδοσιακούς μικροελεγκτές, το ESP32 έχει σχεδιαστεί ειδικά για εφαρμογές IoT, προσφέροντας τον απόλυτο συνδυασμό υψηλής επεξεργαστικής ισχύος και ασύρματης δικτύωσης σε ένα αυτόνομο πακέτο [15].

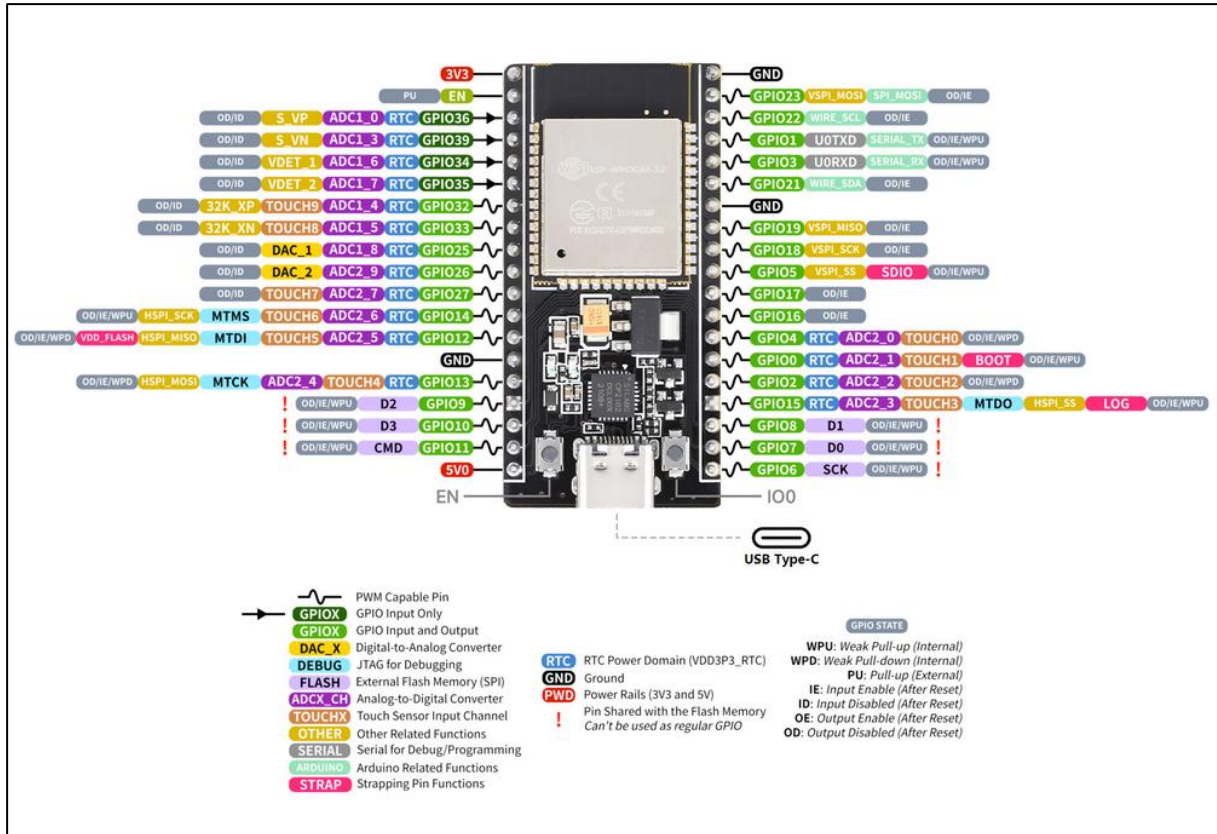
Επεξεργαστική Ισχύς και Μνήμη: Το ESP32 ενσωματώνει έναν διπύρηνο μικροεπεξεργαστή Xtensa 32-bit LX6, ο οποίος μπορεί να χρονιστεί έως και τα 240 MHz. Σε επίπεδο μνήμης, διαθέτει 520 KB εσωτερικής μνήμης SRAM και 448 KB ROM, παρέχοντας άφθονο χώρο για την απρόσκοπτη εκτέλεση της MicroPython, η οποία είναι πιο απαιτητική σε πόρους από τη γλώσσα C++.

Wi-Fi & Bluetooth: Το ισχυρότερο πλεονέκτημα του τσιπ είναι η ενσωματωμένη υποστήριξη 2.4 GHz Wi-Fi (πρωτόκολλα 802.11 b/g/n) και Bluetooth (v4.2 BR/EDR και BLE). Μέσω του Wi-Fi, ο μικροελεγκτής καθίσταται ικανός να δημιουργεί τα απαραίτητα HTTP POST αιτήματα για τη μετάδοση των τηλεμετρικών δεδομένων (JSON) απευθείας στον διαδικτυακό διακομιστή (Django).

Είσοδοι και Έξοδοι: Το τσιπ διαθέτει 34 προγραμματιζόμενες ακίδες (GPIOs) που υποστηρίζουν μια ποικιλία πρωτοκόλλων επικοινωνίας. Περιλαμβάνει ελεγκτές I2C, SPI και UART, οι οποίοι είναι απαραίτητοι για την επικοινωνία με ψηφιακούς αισθητήρες, καθώς και μετατροπείς Αναλογικού σε Ψηφιακό (12-bit SAR ADC) για την ανάγνωση αναλογικών σημάτων.

Κατανάλωση: Δεδομένου ότι το σύστημα ελέγχει μπαταρίες, η ίδια κατανάλωση ενέργειας πρέπει να είναι ελάχιστη. Το ESP32 έχει κατασκευαστεί με τεχνολογία 40 nm χαμηλής ισχύος και διαθέτει προηγμένη διαχείριση ενέργειας με λειτουργίες όπως η Deep-sleep. Σε αυτή την κατάσταση, η κατανάλωση ρεύματος μπορεί να πέσει έως και τα 10 μ A, επιτρέποντας στο σύστημα να εξοικονομεί ενέργεια όταν δεν καταγράφονται κρίσιμες μετρήσεις.

Η λειτουργία των ακροδεκτών του φαίνεται παρακάτω στην Εικόνα 6.1.



Εικόνα 6.1: Ο ESP32 NodeMCU-32S [34]

6.3 Οι Αισθητήρες του Συστήματος

Στην παρούσα κατασκευή, η αξιόπιστη παρακολούθηση του συστήματος και ο υπολογισμός της κατάστασης φόρτισης (State of Charge - SoC) της μπαταρίας, καθιστούν αναγκαία τη συνεχή λήψη μετρήσεων σε πραγματικό χρόνο. Πιο συγκεκριμένα, η αρχιτεκτονική περιλαμβάνει τον ψηφιακό αισθητήρα θερμοκρασίας DS18B20 (μέσω πρωτοκόλλου 1-Wire) και τον προηγμένο αισθητήρα ισχύος INA226 (μέσω πρωτοκόλλου I2C), ο οποίος συνεργάζεται με μια εξωτερική αντίσταση διακλάδωσης (Shunt) για την ασφαλή διαχείριση των υψηλών ρευμάτων του φωτοβολταϊκού πάνελ.

6.3.1 Ο αισθητήρας DS18B20

Ο DS18B20, ο οποίος φαίνεται στην Εικόνα 6.2, είναι ένας αδιάβροχος αισθητήρας που μετράει θερμοκρασία και στο συγκεκριμένο σύστημα, τη θερμοκρασία της μπαταρίας. Το εύρος θερμοκρασίας όπου μετράει είναι από -55°C έως $+125^{\circ}\text{C}$ με ανάλυση 9-12 bits και ακρίβεια $\pm 0.5^{\circ}\text{C}$ (-10°C to $+85^{\circ}\text{C}$). Κάθε αισθητήρας DS18B20 έχει ένα μοναδικό σειριακό αριθμό από 64-bit όπου δίνει την δυνατότητα να συνδεθούν πολλοί μαζί πάνω στο ίδιο "καλώδιο" bus. Η τροφοδοσία κυμαίνεται μεταξύ 3.0V έως 5.5V DC [14].



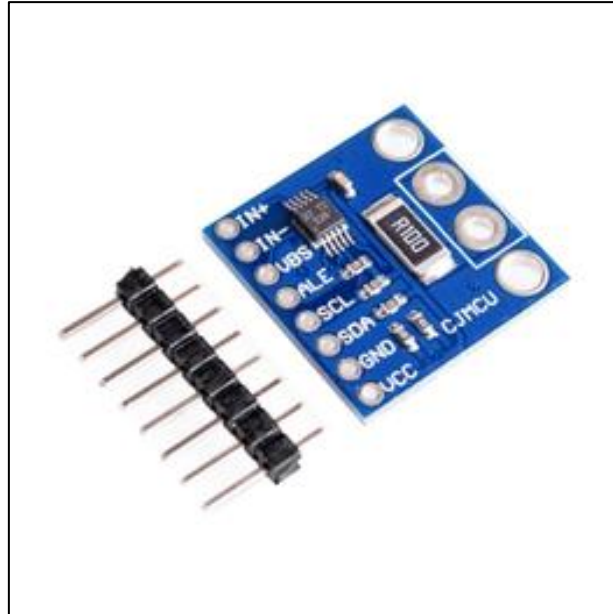
Εικόνα 6.2: Αισθητήρας Θερμοκρασίας DS18B20 [13]

Προδιαγραφές DS18B20:

- Temperature measuring range $-55...125^{\circ}\text{C}$ (-67°F to $+257^{\circ}\text{F}$)
- Temperature measurement accuracy: $\pm 0.5^{\circ}\text{C}$ (-10°C to $+85^{\circ}\text{C}$)
- Working Voltage: 3.0V - 5.5V
- Case TO92
- Mounting THT
- Probe is 7mm in diameter and roughly 26mm long.
- Overall length (including wire) is 100cm.
- Unique 1-Wire® interface requires only one port pin for communication
- Each device has a unique 64-bit serial code stored in an onboard ROM
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Thermometer resolution is user-selectable from 9 to 12 bits

6.3.2 Ο αισθητήρας INA226 (CJMCU-226)

Ο αισθητήρας INA226, που φαίνεται και στην Εικόνα 6.3, είναι ένας εξειδικευμένος ελεγκτής παρακολούθησης ρεύματος διακλάδωσης (shunt) και ισχύος, ο οποίος επικοινωνεί μέσω διεπαφής συμβατής με I2C ή SMBus. Στο παρόν σύστημα IoT, είναι υπεύθυνος για την καταγραφή των ηλεκτρικών δεδομένων που απαιτούνται για τον αλγόριθμο υπολογισμού της κατάστασης φόρτισης (SoC).[16].



Εικόνα 6.3: Ο αισθητήρας INA226 [18]

Προδιαγραφές INA226:

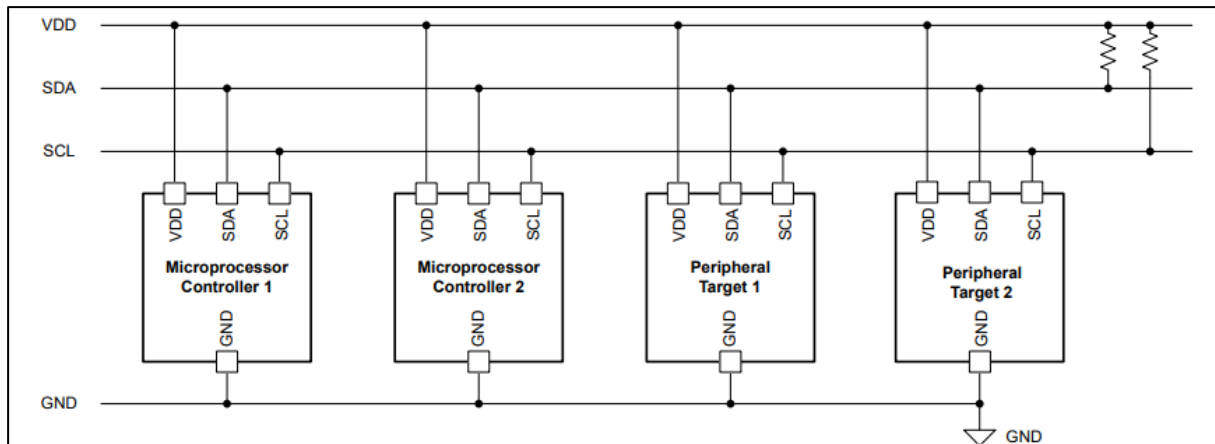
- Senses bus voltages from 0V ~ 36V
- High-side or low-side sensing
- Reports current, voltage, and power
- High accuracy:
- 0.1% gain error(max)
- 10uV offset(max)
- Configurable averaging options
- 12 programmable addresses
- Operates from 2.7V ~ 5.5V power supply
- 10-pin, DGS(VSSOP) package

6.3.3 Το πρωτόκολλο I2C

Η ψηφιακή επικοινωνία μεταξύ των υποσυστημάτων πραγματοποιείται μέσω της διεπαφής I2C, η οποία αποτελεί ενσωματωμένο πρότυπο επικοινωνίας στα χρησιμοποιούμενα υλικά. Το πρωτόκολλο βασίζεται στη χρήση ενός σειριακού διαύλου δύο γραμμών:

- **SCL (Serial Clock Line):** Γραμμή χρονισμού που συγχρονίζει τη μεταφορά των δεδομένων.
- **SDA (Serial Data Line):** Γραμμή για την αμφίδρομη ροή της πληροφορίας.

Η επικοινωνία ακολουθεί την αρχιτεκτονική Κύριου-Δευτερεύοντος (Master-Slave), όπου ο μικροελεγκτής (Master) ελέγχει αποκλειστικά τη γραμμή χρονισμού και την έναρξη των συναλλαγών στο δίαυλο.



Εικόνα 6.4: Τυπική υλοποίηση I2C [19]

Η διαδικασία ανάγνωσης δεδομένων μέσω του πρωτοκόλλου I2C, όπως αποτυπώνεται στη Εικόνα 6.4, ξεκινά με την Κατάσταση Έναρξης (Start Condition), όπου ο Master μεταβάλλει τη στάθμη της γραμμής SDA από High σε Low ενώ η SCL παραμένει High, και συνεχίζεται με τη Διευθυνσιοδότηση (Slave Address Byte), κατά την οποία αποστέλλεται η 7-bit διεύθυνση της συσκευής-στόχου μαζί με το bit κατεύθυνσης (R/W) σε κατάσταση "Εγγραφής" για τον προσδιορισμό του εσωτερικού καταχωρητή, ακολουθούμενη από το σήμα επιβεβαίωσης ACK. Στη συνέχεια, ο Master αποστέλλει τη διεύθυνση του επιθυμητού καταχωρητή (Register Pointer Byte), λαμβάνει εκ νέου επιβεβαίωση ACK και προχωρά σε Επαναλαμβανόμενη Έναρξη (Repeated Start) με το bit κατεύθυνσης σε "Ανάγνωση" για τη μετάβαση στη λήψη δεδομένων χωρίς διακοπή της σύνδεσης. Κατά τη Μεταφορά Δεδομένων, η δευτερεύουσα συσκευή αποστέλλει τα δεδομένα σε μορφή bytes (MSByte και LSByte) με τον Master να επιβεβαιώνει κάθε λήψη, μέχρι τον τελικό Τερματισμό (Stop Condition), όπου μετά τη λήψη του τελευταίου byte ο Master αποστέλλει σήμα No ACK και ολοκληρώνει τη συναλλαγή με σήμα Stop, απελευθερώνοντας τον διάυλο επικοινωνίας [19].

6.4 Το Κύκλωμα Ισχύος

Σε αυτή την ενότητα αναλύεται το κύκλωμα ισχύος, το οποίο αποτελεί τη ραχοκοκαλιά της ενεργειακής υποδομής της κατασκευής. Το συγκεκριμένο κύκλωμα είναι υπεύθυνο για την παραγωγή, τη ρύθμιση, την αποθήκευση και την ελεγχόμενη κατανάλωση της ηλεκτρικής ενέργειας, διασφαλίζοντας την αδιάλειπτη και αυτόνομη λειτουργία του συστήματος IoT. Ειδικότερα, περιγράφονται αναλυτικά τα τεχνικά χαρακτηριστικά και ο ρόλος των βασικών στοιχείων που το απαρτίζουν: το φωτοβολταϊκό πάνελ που συλλέγει την ηλιακή ακτινοβολία, ο ελεγκτής φόρτισης που διαχειρίζεται την ασφαλή μεταφορά της ενέργειας, η μπαταρία μολύβδου που λειτουργεί ως η κύρια μονάδα αποθήκευσης, καθώς και το φορτίο που προσομοιώνει την ενεργειακή απαίτηση του συστήματος. Ο ορθός σχεδιασμός αυτού του υποσυστήματος είναι υψίστης σημασίας, καθώς οι συνθήκες φόρτισης και εκφόρτισης που διαμορφώνει αποτελούν τη βάση για την εξαγωγή αξιόπιστων συμπερασμάτων σχετικά με την Κατάσταση Φόρτισης (SoC) και την Κατάσταση Υγείας (SoH) της μπαταρίας.

6.4.1 Φωτοβολταϊκό Πάνελ 12V (10W)

Η συλλογή της ηλιακής ενέργειας πραγματοποιείται μέσω ενός φωτοβολταϊκού πάνελ ονομαστικής τάσης 12V. Το πάνελ αποτελεί την πρωτεύουσα πηγή τροφοδοσίας του συστήματος, αναλαμβάνοντας τη φόρτιση της μπαταρίας. Τα βασικά τεχνικά χαρακτηριστικά του διαμορφώνονται ως εξής:

- Μέγιστη Ισχύς (Peak Power): 10W

- Ανοχή Παραγωγής (Production Tolerance): $\pm 3\%$
- Ρεύμα Μέγιστης Ισχύος (Maximum Power Current - I_{mp}): 0.572A
- Τάση Μέγιστης Ισχύος (V_{mp}): 17.49V
- Ρεύμα Βραχυκυκλώματος (Short Circuit Current - I_{sc}): 0.61A
- Τάση Ανοιχτού Κυκλώματος (V_{oc}): 21.67V

6.4.2 Μπαταρία Μολύβδου 12V DC

Η αποθήκευση της ηλεκτρικής ενέργειας υλοποιείται μέσω μιας επαναφορτιζόμενης μπαταρίας μολύβδου-οξέος κλειστού τύπου VRLA (Valve-Regulated Lead-Acid). Επιλέχθηκε το μοντέλο Stablepower SP1240, το οποίο προσφέρει αξιοπιστία σε εφαρμογές συνεχούς και εφεδρικής χρήσης.

- Τάση και Χωρητικότητα: 12V, 4.0Ah
- Τάση Κυκλικής Χρήσης (Cycle Use): 14.5V - 15.0V (στους 25°C)
- Τάση Εφεδρικής Χρήσης (Standby Use): 13.5V - 13.8V (στους 25°C)

6.4.3 Ελεγκτής Φόρτισης 30A PWM

Για την ασφαλή διαχείριση της ενέργειας που παράγεται από το φωτοβολταϊκό πάνελ και την προστασία της μπαταρίας, χρησιμοποιείται ένας ρυθμιστής φόρτισης τεχνολογίας PWM (Pulse Width Modulation), ο οποίος φαίνεται και στην Εικόνα 6.5. Ο ελεγκτής παρεμβάλλεται μεταξύ του πάνελ και της μπαταρίας, εξασφαλίζοντας την ορθή καμπύλη φόρτισης.

- Ονομαστική Τάση Λειτουργίας: 12V / 24V (Αυτόματη αναγνώριση)
- Ονομαστικό Ρεύμα (Rated Current): 30A
- Μέγιστη Τάση Εισόδου Φ/Β (Max PV Voltage): 50V
- Μέγιστη Ισχύς Εισόδου: 390W (σε σύστημα 12V) ή 780W (σε σύστημα 24V)



Εικόνα 6.5: Ο Ελεγκτής Φόρτισης [28]

6.4.4 Το Φορτίο (P21W-BA15S-1156)

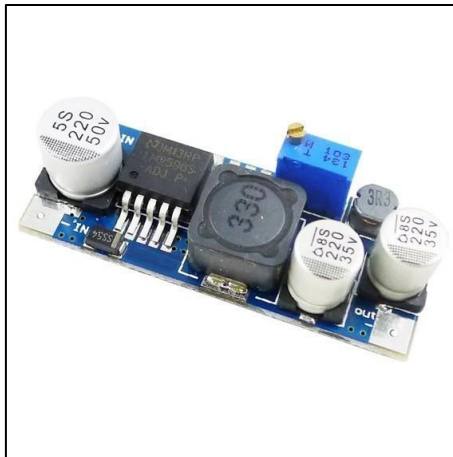
Για την προσομοίωση της κατανάλωσης του συστήματος, χρησιμοποιείται ένας λαμπτήρας πυρακτώσεως τύπου P21W BA15S 1156, σαν αυτούς στην Εικόνα 6.6. Ο συγκεκριμένος λαμπτήρας, ο οποίος χρησιμοποιείται ευρέως σε εφαρμογές αυτοκινήτου, λειτουργεί στα 12V με ονομαστική κατανάλωση ισχύος 21W. Ο φωτισμός που εκπέμπει είναι χρώματος πορτοκαλί (Amber). Η χρήση του ως φορτίο επιτρέπει την ελεγχόμενη και μετρήσιμη πτώση της τάσης της μπαταρίας κατά τις δοκιμές του αλγορίθμου αξιολόγησης (SoH / SoC).



Εικόνα 6.6: Λαμπτήρας P21W BA15S 1156 [28]

6.4.5 Μετατροπέας Τάσης (Step-Down Buck Converter)

Το module LM2596 είναι ένας ρυθμιστής τάσης μεταγωγής (switching regulator) τύπου υποβιβασμού (buck converter) [17] και φαίνεται στην Εικόνα 6.7. Στο παρόν σύστημα IoT, αναλαμβάνει τον κρίσιμο ρόλο της διαχείρισης τροφοδοσίας, παρέχοντας ασφαλή και σταθερή τάση στον μικροελεγκτή ESP32 αντλώντας ενέργεια απευθείας από την μπαταρία του φωτοβολταϊκού (12V - 14.4V) σταθερά στα 5V.



Εικόνα 6.7: LM2596 [19]

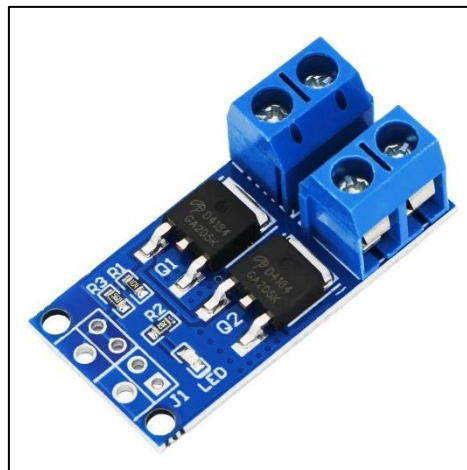
Προδιαγραφές LM2596:

- Input voltage: 4.5-40V
- Output Voltage: 1.5-35V (adjustable)
- Output current: Rated current 2A, maximum 3A (additional heat sink is required)
- Conversion efficiency: Up to 92%
- Potentiometer adjustment direction: Clockwise (increase), Anti-clockwise (decrease)
- Switching frequency: 150KHz
- Rectifier: Non-synchronous rectification
- Module Properties: Non-isolated step-down module
- Short circuit protection: Current limiting, since the recovery
- Operating temperature: -40 °C to +85 °C (output power 10W or less)
- Full load temperature rise: 40 °C
- Dimensions: 52×20×14 mm (with potentiometer)

6.4.6 Μονάδα Ελέγχου Φορτίου (Dual High-Power MOSFET)

Για τον ηλεκτρονικό έλεγχο της ενεργοποίησης και απενεργοποίησης του φορτίου από τον μικροελεγκτή, χρησιμοποιείται μια πλακέτα οδήγησης διπλού MOSFET υψηλής ισχύος (Dual High-Power MOSFET Trigger Switch Drive Module). Το εν λόγω στοιχείο λειτουργεί ως ψηφιακός διακόπτης, επιτρέποντας στον μικροελεγκτή (που λειτουργεί στα 3.3V) να ελέγχει με ασφάλεια τις μεγάλες απαιτήσεις ρεύματος του φορτίου στα 12V.

- **Δομή:** Αξιοποιεί δύο παράλληλα συνδεδεμένα MOSFET, προσφέροντας χαμηλότερη εσωτερική αντίσταση (R_{ds_on}), υψηλότερη αγωγιμότητα ρεύματος και εξαιρετική αντοχή ισχύος (έως 400W).
- **Τάση και Ρεύμα Λειτουργίας:** Υποστηρίζει τάση εισόδου από 5V έως 36V DC και επιτρέπει τη διέλευση συνεχούς ρεύματος 15A σε θερμοκρασία δωματίου (με μέγιστο όριο τα 30A).
- **Πηγή Σήματος (Trigger Source):** Ελέγχεται από ψηφιακά σήματα στάθμης 3.3V - 20V, καθιστώντας το άμεσα συμβατό με τα GPIOs του ESP32. Υποστηρίζει επίσης διαμόρφωση εύρους παλμού (PWM) σε συχνότητες 0 - 20KHz, προσφέροντας τη δυνατότητα ρύθμισης της φωτεινότητας του λαμπτήρα.
- **Θερμοκρασία Λειτουργίας:** -40°C έως 85°C, με απεριόριστη διάρκεια ζωής στους κύκλους μεταγωγής.



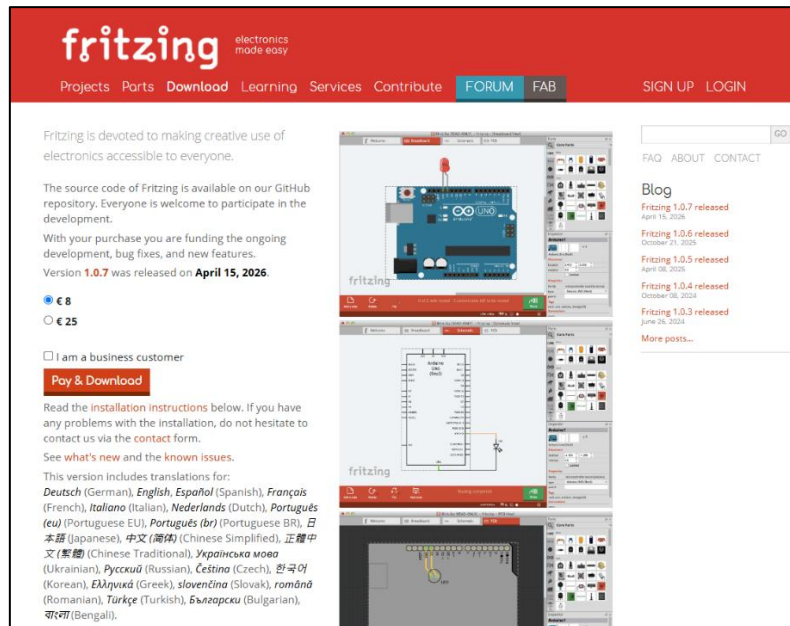
Εικόνα 6.8: Το MOSFET [28]

6.5 Σχεδιασμός και Συνδεσμολογία Κυκλώματος

Στην παρούσα ενότητα παρουσιάζεται η ολοκληρωμένη αρχιτεκτονική του υλικού (hardware), από τον ψηφιακό σχεδιασμό έως την τελική φυσική υλοποίηση της πειραματικής διάταξης. Η διαδικασία αυτή αποτελεί τον πυρήνα του Επιπέδου Αντίληψης, καθώς εξασφαλίζει την ασφαλή και αξιόπιστη διασύνδεση των αισθητήρων, του μικροελεγκτή και των διατάξεων ισχύος.

6.5.1 Το fritzing

Για τον σχεδιασμό του ηλεκτρονικού κυκλώματος της παρούσας εργασίας αξιοποιήθηκε το λογισμικό Fritzing. Η εγκατάστασή του πραγματοποιήθηκε μέσω της επίσημης ιστοσελίδας (<https://fritzing.org/download/>), διαδικασία που απεικονίζεται στην Εικόνα 6.9. Συγκεκριμένα, χρησιμοποιήθηκε η έκδοση 1.0.7 (ημερομηνία κυκλοφορίας: 15 Απριλίου 2026), με το κόστος απόκτησης της άδειας να ανέρχεται στα 8€.



Εικόνα 6.9: Εγκατάσταση Fritzing [20]

Το Fritzing αποτελεί ένα ευρέως διαδεδομένο πρόγραμμα ανοιχτού κώδικα, το οποίο εστιάζει στη δημιουργία σχηματικών διαγραμμάτων και τον σχεδιασμό τυπωμένων κυκλωμάτων (PCBs). Διακρίνεται για το φιλικό του γραφικό περιβάλλον, το οποίο καθιστά τη χρήση του ιδιαίτερα προσιτή και εύκολη, ακόμη και για χρήστες χωρίς μεγάλη εμπειρία στον τομέα.

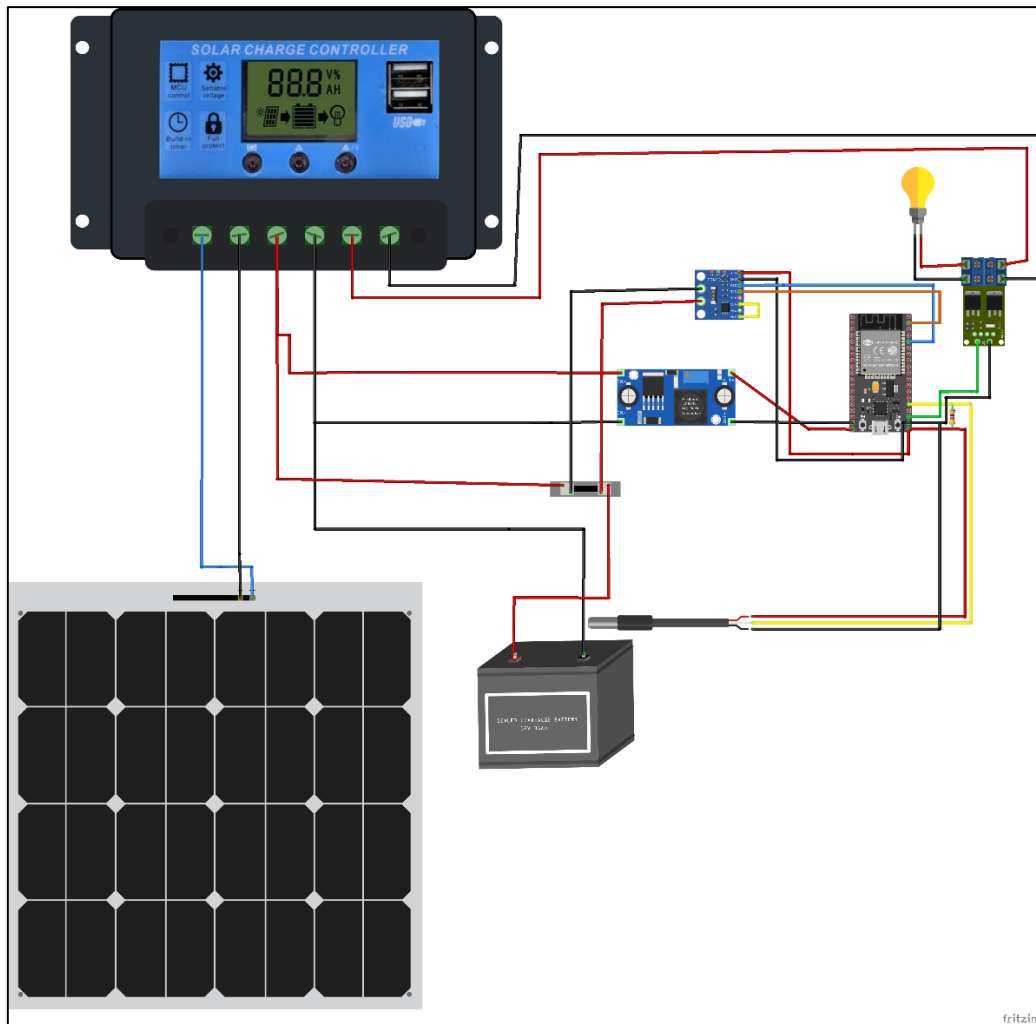
Οι κύριες λειτουργίες του προγράμματος περιλαμβάνουν:

- **Σχεδίαση Τυπωμένων Κυκλωμάτων (PCB):** Παρέχει ένα ευέλικτο περιβάλλον όπου ο χρήστης μπορεί εύκολα να τοποθετήσει τα ηλεκτρονικά εξαρτήματα και να ορίσει τις συνδέσεις των ακροδεκτών τους.
- **Ανάπτυξη Σχηματικών Διαγραμμάτων:** Διαθέτει εξειδικευμένα εργαλεία για την ακριβή οπτική αποτύπωση των ηλεκτρονικών στοιχείων και της καλωδίωσης ενός κυκλώματος.
- **Υποστήριξη από την Κοινότητα:** Διατηρεί μια πολύ ενεργή βάση χρηστών, επιτρέποντας την ανταλλαγή γνώσεων και σχεδίων. Οι χρήστες μπορούν να κατεβάσουν έτοιμα σχηματικά και PCBs ή να μοιραστούν τα δικά τους.
- **Επεκτασιμότητα και Προσαρμογή:** Προσφέρει τη δυνατότητα δημιουργίας και εισαγωγής νέων, προσαρμοσμένων εξαρτημάτων, καλύπτοντας έτσι τις εξειδικευμένες απαιτήσεις κάθε σχεδίου.

Συμπερασματικά, το Fritzing συνιστά ένα εξαιρετικά πρακτικό και ευέλικτο εργαλείο, ιδανικό τόσο για ερασιτεχνική απασχόληση όσο και για επαγγελματίες του χώρου των ηλεκτρονικών που αναζητούν μια αξιόπιστη λύση για τον σχεδιασμό κυκλωμάτων [20].

6.5.2 Το Ηλεκτρονικό Κύκλωμα

Μέσω του Fritzing, δημιουργείται το Ηλεκτρονικό Κύκλωμα το οποίο φαίνεται παρακάτω στο Σχέδιο 6.1.



Σχέδιο 6.1: Το Ηλεκτρονικό Κύκλωμα

Αναλύοντας το Ηλεκτρονικό Κύκλωμα:

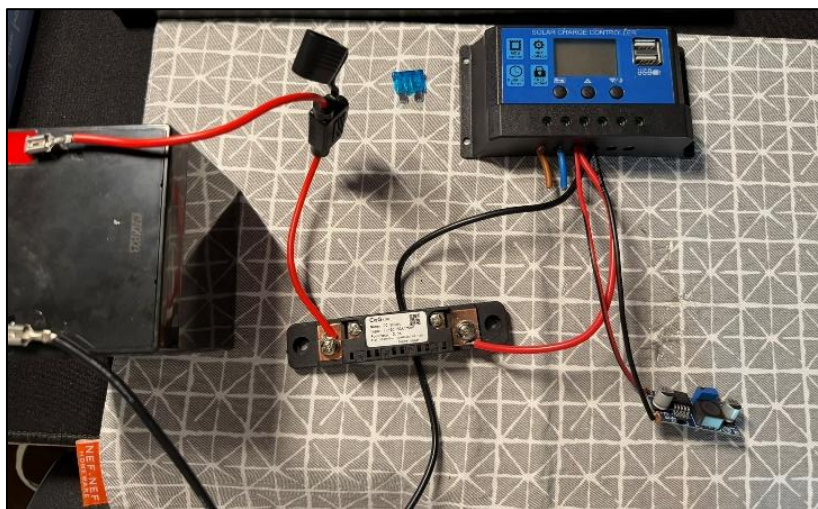
- Το φωτοβολταϊκό πάνελ 10W συνδέεται απευθείας στους αντίστοιχους ακροδέκτες εισόδου του ρυθμιστή φόρτισης PWM. Η έξοδος του ρυθμιστή οδηγείται στους πόλους της μπαταρίας μολύβδου VRLA 12V. Από τους πόλους της μπαταρίας δρομολογείται στην είσοδο του μετατροπέα υποβιβασμού τάσης LM2596. Ο μετατροπέας αυτός ρυθμίζεται μέσω του ενσωματωμένου ποτενσιόμετρου ώστε να αποδίδει σταθερή τάση 5V στην έξοδό του, τροφοδοτώντας με απόλυτη ασφάλεια τον μικροελεγκτή ESP32 μέσω του ακροδέκτη εισόδου τάσης ($V_{in} / 5V$).
- Η εξωτερική αντίσταση διακλάδωσης (Shunt CG FL-2C 50A) παρεμβάλλεται εν σειρά στο κύκλωμα ισχύος για τη διέλευση του ρεύματος. Οι ακροδέκτες μέτρησης του shunt (συνδέονται στις αναλογικές εισόδους IN+ και IN- του αισθητήρα INA226. Η ψηφιακή επικοινωνία του INA226 με τον μικροελεγκτή επιτυγχάνεται μέσω του πρωτοκόλλου I2C, συνδέοντας τους ακροδέκτες δεδομένων (SDA) και χρονισμού (SCL) στις αντίστοιχες θύρες του ESP32 (GPIO 21 και GPIO 22 αντίστοιχα). Η τροφοδοσία του αισθητήρα παρέχεται από τον ακροδέκτη 3.3V του μικροελεγκτή.
- Ο αισθητήρας θερμοκρασίας DS18B20 τοποθετείται σε άμεση επαφή με το περίβλημα της μπαταρίας για την ακριβή καταγραφή της θερμικής της κατάστασης. Ο ακροδέκτης δεδομένων συνδέεται σε μια ψηφιακή είσοδο του ESP32 (GPIO 4). Για τη σταθεροποίηση της επικοινωνίας μέσω του πρωτοκόλλου 1-Wire, χρησιμοποιείται μια αντίσταση pull-up 4.7kΩ, η οποία γεφυρώνει τη γραμμή δεδομένων με τη γραμμή τροφοδοσίας 3.3V του μικροελεγκτή.

- Ο μικροελεγκτής αναλαμβάνει τον δυναμικό έλεγχο της εκφόρτισης (λαμπτήρας P21W) μέσω της πλακέτας οδήγησης διπλού MOSFET. Ένας ψηφιακός ακροδέκτης εξόδου του ESP32 (GPIO 15) συνδέεται στην είσοδο Trigger / PWM του MOSFET. Όταν ο εσωτερικός αλγόριθμος τοπικού ελέγχου (Edge BMS) του μικροελεγκτή κρίνει ότι η μπαταρία βρίσκεται σε ασφαλή επίπεδα λειτουργίας, στέλνει ένα ψηφιακό σήμα λογικού '1' (3.3V) στο MOSFET. Αυτό έχει ως αποτέλεσμα να κλείσει το κύκλωμα ισχύος και να επιτρέψει τη ροή ρεύματος προς τον λαμπτήρα. Σε περίπτωση υπέρβασης των προκαθορισμένων ορίων, όπως πτώση τάσης κάτω από 11.5V ή υπερθέρμανση, το σήμα μηδενίζεται, αποκόπτοντας ακαριαία το φορτίο για την προστασία του συστήματος.

6.5.3 Η Κατασκευή

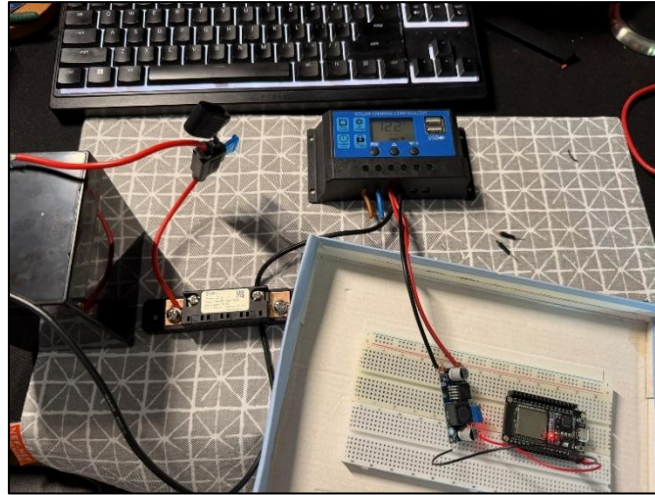
Η τελική φάση του σχεδιασμού του υλικού αφορά τη μετάβαση από το θεωρητικό ψηφιακό μοντέλο στη φυσική πειραματική διάταξη. Η πρακτική υλοποίηση της κατασκευής και η διασύνδεση των υποσυστημάτων πραγματοποιήθηκε με μεθοδικότητα, ακολουθώντας τα εξής διακριτά βήματα υλοποίησης:

- Βήμα 1ο: Σύνδεση Γραμμής Ισχύος, Ρυθμιστή και Shunt
Αρχικά υλοποιήθηκε το πρωτεύον κύκλωμα ισχύος του συστήματος. Ο ρυθμιστής φόρτισης PWM τοποθετήθηκε ως κεντρικός κόμβος διαχείρισης και συνδέθηκε στους πόλους της μπαταρίας μολύβδου VRLA 12V (4Ah), παρεμβάλλοντας μια ασφάλεια συνεχούς ρεύματος (DC inline fuse) στον θετικό αγωγό για προστασία από βραχυκυκλώματα. Στην κύρια γραμμή επιστροφής παρεμβλήθηκε εν σειρά η εξωτερική αντίσταση διακλάδωσης (Shunt 50A/75mV), ενώ οι ακροδέκτες εισόδου του μετατροπέα τάσης LM2596 συνδέθηκαν απευθείας στις γραμμές του ρυθμιστή φόρτισης, όπως αποτυπώνεται στην Εικόνα 6.10.

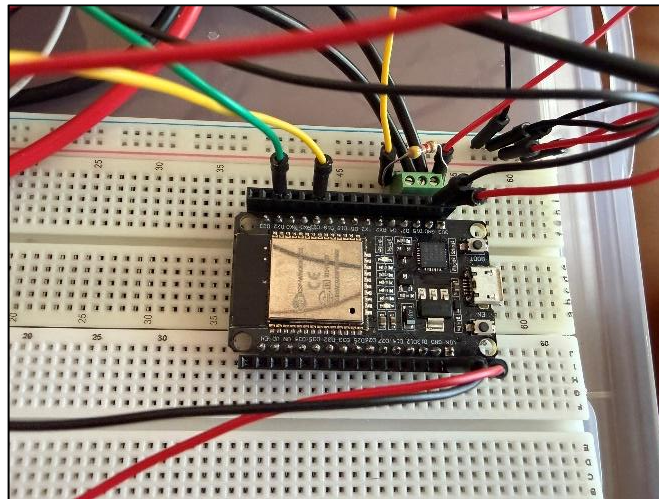


Εικόνα 6.10: Σύνδεση μπαταρίας, ρυθμιστή, ασφάλειας και shunt

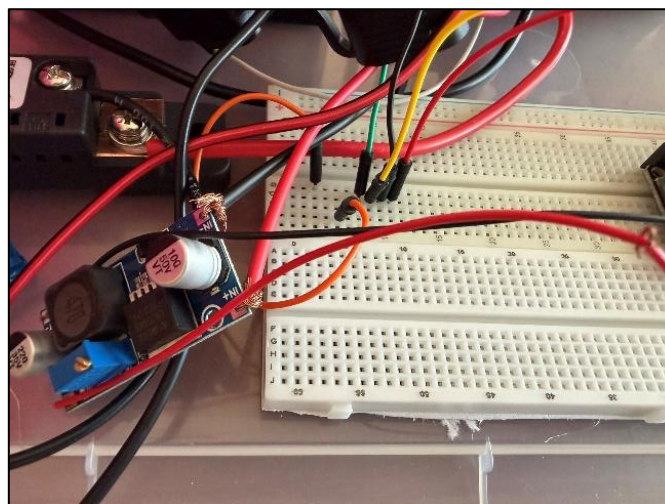
- Βήμα 2ο: Πρωτοτυποποίηση σε Breadboard και Έλεγχος Τροφοδοσίας 5V
Πριν από οποιαδήποτε μόνιμη συγκόλληση, τα επιμέρους στοιχεία συνδέθηκαν προσωρινά σε πλακέτα δοκιμών (Breadboard) προκειμένου να επαληθευτεί η ορθή λειτουργία του κυκλώματος και να ελεγχθεί η ασφαλής παροχή τάσης στον μικροελεγκτή (Εικόνα 6.11). Με τη χρήση ψηφιακού πολυμέτρου, το ενσωματωμένο ποτενσιόμετρο του μετατροπέα υποβιβασμού LM2596 ρυθμίστηκε ώστε να παρέχει σταθερή τάση εξόδου αυστηρά στα 5V. Η σταθεροποιημένη τάση οδηγήθηκε στους ακροδέκτες Vin και GND του ESP32, επιβεβαιώνοντας την ομαλή εκκίνηση της πλακέτας και αποτρέποντας τυχόν καταστροφή της από υπέρταση (Εικόνες 6.12 και 6.13).



Εικόνα 6.11: Γενική επισκόπηση της δοκιμαστικής διάταξης σε breadboard

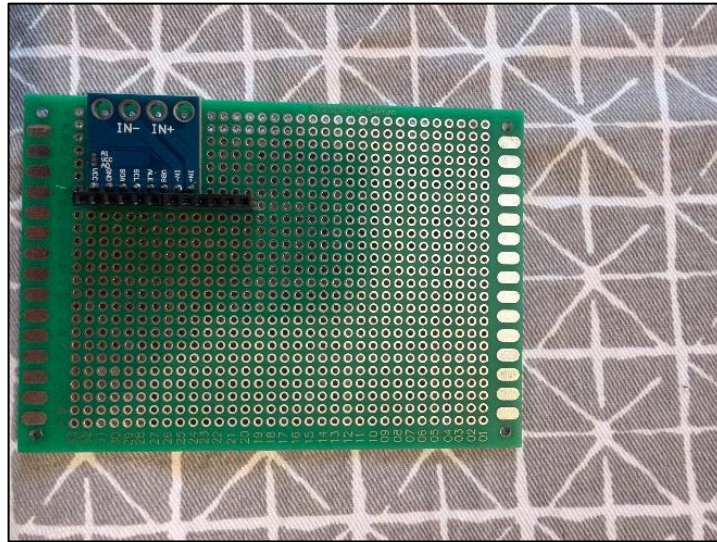


Εικόνα 6.12: Δοκιμή ESP32 στο breadboard



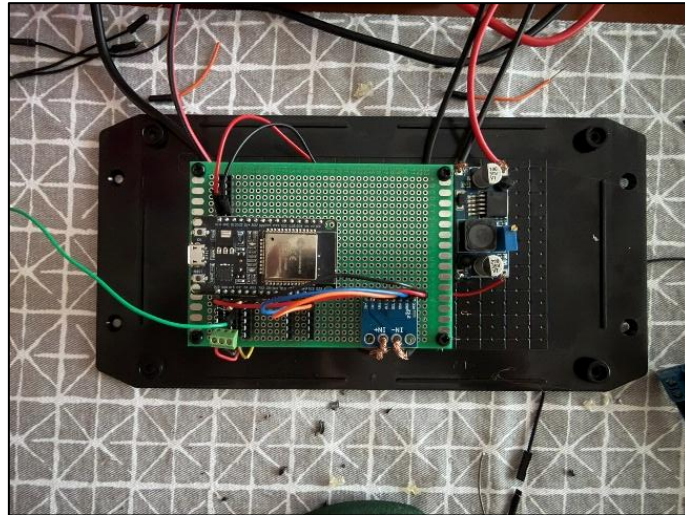
Εικόνα 6.13: Ρύθμιση τροφοδοσίας 5V LM2596

- **Βήμα 3ο: Προετοιμασία και Συγκόλληση Διάτρητης Πλακέτας (Perfboard)**
Μετά την επιβεβαίωση της σωστής λειτουργίας των κυκλωμάτων στο breadboard, σχεδιάστηκε η μόνιμη πλακέτα διασύνδεσης. Χρησιμοποιήθηκε διάτρητη πλακέτα εποξειδικής ρητίνης (perfboard), στην οποία συγκολλήθηκαν κατάλληλοι θηλυκοί ακροδέκτες (headers) για την υποδοχή του ESP32 και του αισθητήρα INA226 (Εικόνα 6.14). Η προσέγγιση επιτρέπει την ταχεία αφαίρεση ή αντικατάσταση των εξαρτημάτων, προστατεύοντάς τα παράλληλα από θερμική καταπόνηση κατά τις κολλήσεις. Στην ίδια πλακέτα ενσωματώθηκε και η αντίσταση pull-up $4.7k\Omega$ για τη γραμμή δεδομένων του αισθητήρα θερμοκρασίας DS18B20.

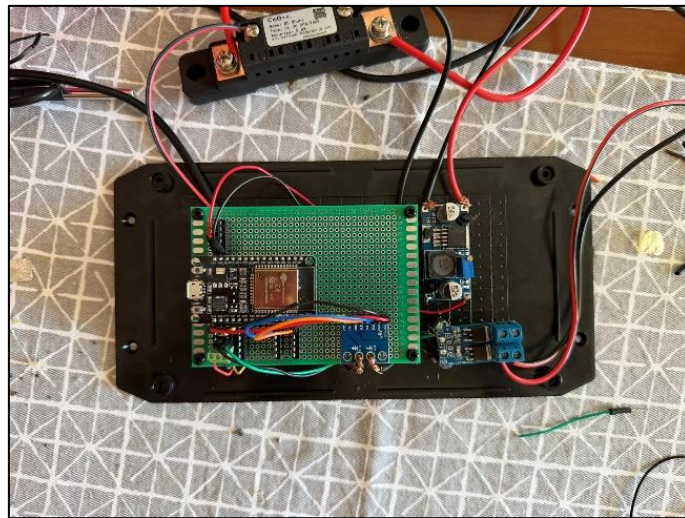


Εικόνα 6.14: Συγκόλληση headers στη διάτρητη

- **Βήμα 4ο: Μηχανική Συναρμολόγηση και Σταθεροποίηση Υποσυστημάτων**
Στη συνέχεια, όλα τα επιμέρους υποσυστήματα (η διάτρητη πλακέτα με τον ESP32 και τον INA226, ο μετατροπέας LM2596, η μονάδα οδήγησης διπλού MOSFET και οι κλέμες τροφοδοσίας) μεταφέρθηκαν και σταθεροποιήθηκαν μηχανικά επάνω σε μια άκαμπτη μαύρη βάση στήριξης μέσω πλαστικών αποστατών και κοχλιών (Εικόνα 6.15). Ολοκληρώθηκαν οι καλωδιώσεις των σημάτων επικοινωνίας: ο δίαυλος I2C (SDA/SCL) για τη σύνδεση του INA226, η γραμμή 1-Wire για τον DS18B20 και η ψηφιακή έξοδος GPIO 15 που ελέγχει την πύλη (Trigger) του MOSFET, ολοκληρώνοντας τη συναρμολόγηση του κεντρικού πίνακα ελέγχου (Εικόνα 6.16).

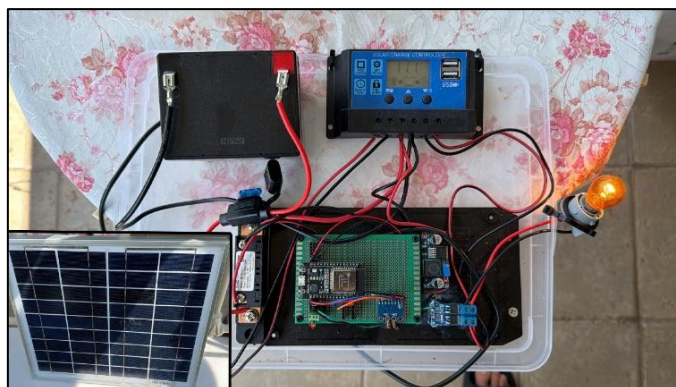


Εικόνα 6.15: Σταθεροποίηση μονάδων στη βάση



Εικόνα 6.16: Τελική διάταξη βάσης πριν τη δοκιμή

- Βήμα 5ο: Τελική Ολοκλήρωση, Σύνδεση Φορτίου και Πειραματική Διάταξη
Στο τελικό στάδιο, η βάση στήριξης των ηλεκτρονικών τοποθετήθηκε εντός προστατευτικού διαφανούς κυτίου για ασφάλεια. Ο αισθητήρας θερμοκρασίας DS18B20 στερεώθηκε σε άμεση μηχανική και θερμική επαφή με το εξωτερικό περίβλημα της μπαταρίας. Συνδέθηκαν το φωτοβολταϊκό πάνελ 10W στους ακροδέκτες εισόδου του ρυθμιστή φόρτισης και ο λαμπτήρας πυρακτώσεως P21W στην έξοδο ισχύος του κυκλώματος MOSFET (Εικόνα 6.17). Με τον τρόπο αυτό ολοκληρώθηκε η πλήρης πειραματική διάταξη, καθιστώντας το σύστημα απόλυτα λειτουργικό για τις μετρήσεις και τα πειράματα εκφόρτισης.



Εικόνα 6.17: Η τελική μορφή της πειραματικής διάταξης

Το συνολικό κόστος της κατασκευής, μαζί με την αγορά της άδειας του fritzing υπολογίζεται περίπου στα 125 € και φαίνεται αναλυτικά στον Πίνακα 6.1.

Πίνακας 6.1: Κοστολόγιο του συστήματος

Προϊόντα	Ενδεικτικό Κόστος
Μικροελεγκτής ESP32 NodeMCU-32S	12€
Αισθητήρας Θερμοκρασίας DS18B20	4€
Αισθητήρας Τάσης/Ρεύματος INA226	12€
Αντίσταση Διακλάδωσης (Shunt 50A)	7€
Μονάδα Διπλού MOSFET	4€
Φωτοβολταϊκό Πάνελ 12V (10W)	Δανεισμός (~20€)
Ελεγκτής Φόρτισης PWM 30A	Δανεισμός (~15€)
Μπαταρία Μολύβδου VRLA 12V 4.0Ah	12€
Λαμπτήρας (Φορτίο) & Διάφορα (Καλώδια, Breadboard, LM2596)	30€
Fritzing	8€

Κεφάλαιο 7ο: Επίπεδο Δικτύου (Network Layer)

7.1 Εισαγωγή

Στο παρόν κεφάλαιο αναλύεται το επίπεδο δικτύου/μετάδοσης -Network Layer- το οποίο αποτελεί τον ενδιάμεσο κρίκο διασύνδεσης μεταξύ του φυσικού επιπέδου αντίληψης και του διαδικτυακού επιπέδου εφαρμογής (Django Backend). Ο κύριος ρόλος αυτού του επιπέδου είναι η ασφαλής, αξιόπιστη και αδιάλειπτη μεταφορά των τηλεμετρικών δεδομένων προς τον κεντρικό διακομιστή. Ειδικότερα, εξετάζεται η διαδικασία σύνδεσης του συστήματος στο τοπικό ασύρματο δίκτυο (Wi-Fi), τα πρωτόκολλα που επιλέχθηκαν για τη δρομολόγηση της πληροφορίας, καθώς και οι στρατηγικές διαχείρισης σφαλμάτων που υλοποιήθηκαν στον κώδικα για την αντιμετώπιση απωλειών σύνδεσης.

7.2 Σύνδεση στο Δίκτυο Wi-Fi

Η ασύρματη δικτύωση του συστήματος επιτυγχάνεται μέσω της ενσωματωμένης μονάδας Wi-Fi του μικροελεγκτή ESP32, η οποία υποστηρίζει τα πρότυπα 802.11 b/g/n. Η υλοποίηση της σύνδεσης πραγματοποιείται σε επίπεδο λογισμικού (MicroPython) μέσω της ενσωματωμένης βιβλιοθήκης network, ρυθμίζοντας τον μικροελεγκτή σε κατάσταση σταθμού (Station Mode - STA_IF) και φαίνεται στην Εικόνα 7.1.

Βασική αρχιτεκτονική επιλογή κατά την ανάπτυξη του δικτυακού κώδικα αποτέλεσε η ασφάλεια των διαπιστευτηρίων. Για την αποτροπή έκθεσης ευαίσθητων δεδομένων, το όνομα του δικτύου (SSID) και ο κωδικός πρόσβασης (Password) δεν δηλώνονται απευθείας στον κύριο κώδικα. Αντιθέτως, εισάγονται με ασφάλεια μέσω ενός ανεξάρτητου αρχείου διαμόρφωσης (secrets.py), διασφαλίζοντας την προστασία του τοπικού δικτύου σε περίπτωση δημοσιοποίησης του πηγαίου κώδικα.

```
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print("Σύνδεση στο Wi-Fi...", end="")
        wlan.connect(secrets.WIFI_SSID, secrets.WIFI_PASS)
        timeout = 20
        while not wlan.isconnected() and timeout > 0:
            time.sleep(1)
            print(".", end="")
            timeout -= 1
        if not wlan.isconnected():
            print("\n[ERROR] Αποτυχία σύνδεσης στο Wi-Fi. Επανεκκίνηση...")
            import machine
            machine.reset()
        print(f"\n[NETWORK] Επιτυχής σύνδεση! IP: {wlan.ifconfig()[0]}")
    return wlan
```

Εικόνα 7.1: Συνάρτηση αρχικοποίησης και σύνδεσης στο ασύρματο δίκτυο

7.3 Πρωτόκολλο Μεταφοράς Δεδομένων (HTTP POST)

Για την ασφαλή και αποδοτική μετάδοση των τηλεμετρικών δεδομένων από τον μικροελεγκτή προς τη διαδικτυακή πλατφόρμα, επιλέχθηκε το πρωτόκολλο HTTP (Hypertext Transfer Protocol). Ειδικότερα, χρησιμοποιείται η μέθοδος POST, η οποία ενδείκνυται για την αποστολή νέων δεδομένων προς αποθήκευση σε έναν απομακρυσμένο διακομιστή (Server API).

Τα δεδομένα δομούνται και κωδικοποιούνται σε μορφή JSON (JavaScript Object Notation). Το πρότυπο JSON προτιμήθηκε καθώς αποτελεί το κυρίαρχο βιομηχανικό πρότυπο για την ανταλλαγή δεδομένων. Προσφέρει μια ελαφριά δομή που δεν επιβαρύνει το περιορισμένο εύρος ζώνης ενός IoT κόμβου, ενώ

επιτρέπει την οργάνωση της πληροφορίας σε ζεύγη κλειδιού-τιμής όπως στην Εικόνα 7.2, διευκολύνοντας την άμεση αποκωδικοποίηση από το Django backend.

```
payload = {  
    "mac_address": MAC_ADDRESS,  
    "voltage": real_voltage,  
    "current": real_current,  
    "temperature": temp_c  
}
```

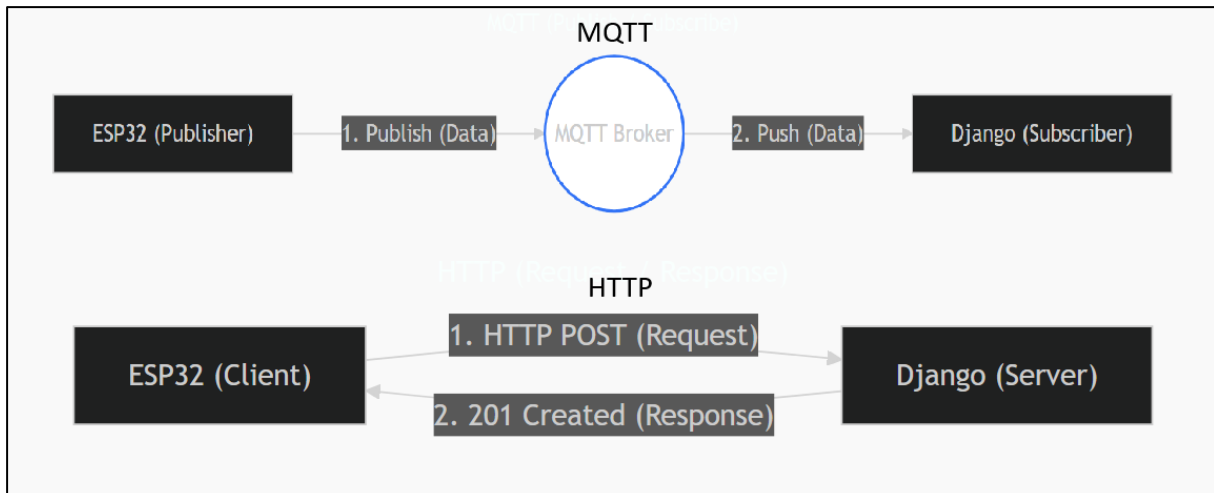
Εικόνα 7.2: Δομή και κωδικοποίηση των τηλεμετρικών δεδομένων σε μορφή JSON

7.3.1 HTTP POST έναντι MQTT

Στο πλαίσιο του οικοσυστήματος του Διαδικτύου των Πραγμάτων (Internet of Things, IoT), η επιλογή του κατάλληλου πρωτοκόλλου επικοινωνίας για τη μετάδοση δεδομένων από το Επίπεδο Αντίληψης (αισθητήρες/μικροελεγκτής) προς το Επίπεδο Εφαρμογής (διακομιστής/βάση δεδομένων) συνιστά κρίσιμη αρχιτεκτονική απόφαση. Μεταξύ των ευρύτερα διαδεδομένων πρωτοκόλλων για τον σκοπό αυτό συγκαταλέγονται το HTTP (Hypertext Transfer Protocol) και το MQTT (Message Queuing Telemetry Transport).

Το MQTT αποτελεί, κατ' εξοχήν, το βιομηχανικό πρότυπο για εφαρμογές IoT. Δεν βασίζεται στο παραδοσιακό μοντέλο Πελάτη–Διακομιστή, αλλά στο αρχιτεκτονικό υπόδειγμα Δημοσίευσης–Εγγραφής (Publish/Subscribe). Η λειτουργία του προϋποθέτει, συνεπώς, την ύπαρξη ενός ενδιάμεσου διακομιστή, γνωστού ως «Μεσίτη» (Broker – π.χ. Eclipse Mosquitto), ο οποίος αναλαμβάνει τη δρομολόγηση των μηνυμάτων μεταξύ των συμμετεχόντων κόμβων. Το πρωτόκολλο διακρίνεται για το ιδιαίτερα περιορισμένο μέγεθος της επικεφαλίδας του (header overhead) και τη βέλτιστη διαχείριση της ενεργειακής κατανάλωσης, χαρακτηριστικά που το καθιστούν ιδανικό για δίκτυα περιορισμένου εύρους ζώνης (bandwidth) ή ασταθούς συνδεσιμότητας.

Αντιθέτως, το πρωτόκολλο HTTP, βασίζεται στο κλασικό μοντέλο Πελάτη–Διακομιστή (Client/Server) και στη λογική Αιτήματος–Απόκρισης (Request/Response). Ο μικροελεγκτής, ενεργώντας ως Πελάτης (Client), εγκαθιστά μια σύγχρονη σύνδεση TCP, αποστέλλει τα δεδομένα μέσω της μεθόδου POST προς ένα συγκεκριμένο τελικό σημείο (API Endpoint) και αναμένει τη σχετική επιβεβαίωση (HTTP Status Code, π.χ. 201 Created) πριν από τον τερματισμό της σύνδεσης. Η θεμελιώδης διαφορά μεταξύ των δύο αυτών αρχιτεκτονικών προσεγγίσεων, δηλαδή του αποκεντρωμένου μοντέλου Publish/Subscribe του MQTT και του παραδοσιακού Request/Response του HTTP, αποτυπώνεται σχηματικά στην Εικόνα 7.3.



Εικόνα 7.3: MQTT έναντι HTTP

Παρότι το πρωτόκολλο MQTT θεωρείται, κατά κανόνα, «ελαφρύτερο» και προτιμάται ευρέως σε εφαρμογές IoT, στο πλαίσιο της παρούσας διπλωματικής εργασίας επιλέχθηκε συνειδητά η αξιοποίηση του πρωτοκόλλου HTTP POST, σε συνδυασμό με αρχιτεκτονική τύπου REST API, για τους ακόλουθους λόγους:

- 1. Απλοποίηση της Αρχιτεκτονικής Συστήματος:** Η υιοθέτηση του HTTP κατέστησε δυνατή την απευθείας επικοινωνία του μικροελεγκτή ESP32 με το backend υλοποιημένο σε Django. Αντίθετα, η επιλογή του MQTT θα προϋπέθετε την εγκατάσταση, τη διαμόρφωση και τη διαρκή συντήρηση πρόσθετης υποδομής λογισμικού (MQTT Broker), καθώς και την ανάπτυξη ενός ενδιάμεσου σεναρίου (daemon script) στον διακομιστή, το οποίο θα παρέμενε σε συνεχή ακρόαση των μηνυμάτων του broker προκειμένου να τα καταχωρεί στη βάση δεδομένων PostgreSQL.
- 2. Εγγενής Υποστήριξη από το Πλαίσιο Django:** Το framework Django είναι σχεδιασμένο και δομημένο αποκλειστικά γύρω από το πρωτόκολλο HTTP. Η ανάπτυξη ενός API ικανού να δέχεται πακέτα δεδομένων σε μορφή JSON συνιστά, συνεπώς, μια φυσική, ιδιαίτερα βελτιστοποιημένη και ασφαλή διαδικασία εντός του οικοσυστήματος της Python, αξιοποιώντας τους ενσωματωμένους μηχανισμούς διαχείρισης σφαλμάτων και ταυτοποίησης (authentication) που παρέχει το πλαίσιο.
- 3. Ανεξαρτησία Κατάστασης (Statelessness):** Η φιλοσοφία σχεδιασμού του υλοποιηθέντος Edge BMS προβλέπει ότι ο μικροελεγκτής λαμβάνει μετρήσεις, προβαίνει σε τοπικές αποφάσεις και, εν συνεχεία, ενημερώνει το σύστημα cloud. Υπό αυτές τις συνθήκες, η διατήρηση μόνιμα ανοικτής σύνδεσης δικτύου δεν κρίθηκε απαραίτητη. Κάθε πακέτο JSON μεταφέρει αυτόνομα το σύνολο των απαιτούμενων πληροφοριών (διεύθυνση MAC, τάση, ρεύμα, θερμοκρασία), ικανοποιώντας πλήρως την αρχή του statelessness που διέπει την αρχιτεκτονική REST.

Συμπερασματικά, ενώ το MQTT αποτελεί τη βέλτιστη λύση για τεράστιας κλίμακας εγκαταστάσεις (Massive IoT) με χιλιάδες ταυτόχρονους κόμβους σε περιορισμένα δίκτυα, το πρωτόκολλο HTTP προσφέρει μια πολύ πιο άμεση, εύκολη στη διαχείριση και απόλυτα συμβατή λύση για τη διασύνδεση ενός σταθμού παρακολούθησης φωτοβολταϊκών με ένα σύγχρονο Web Framework.

7.4 Διαδικασία Αποστολής και Συγχρονισμού

Η επικοινωνία του μικροελεγκτή με τον τοπικό διακομιστή πραγματοποιείται μέσω της βιβλιοθήκης `urequests` της MicroPython. Η διαδικασία του συγχρονισμού σχεδιάστηκε με κύριο γνώμονα την ευρωστία (robustness) και τη διαχείριση των περιορισμένων πόρων του ESP32.

Δύο κρίσιμα ζητήματα που αντιμετωπίστηκαν σε αυτό το επίπεδο είναι η ανοχή σε σφάλματα (fault tolerance) και η διαρροή μνήμης (memory leak). Η δικτυακή επικοινωνία περιβάλλεται από μηχανισμούς διαχείρισης εξαιρέσεων, ώστε τυχόν καθυστερήσεις του δικτύου να μην προκαλούν κατάρρευση του συστήματος. Επιπλέον, επειδή οι μικροελεγκτές διαθέτουν ελάχιστη μνήμη RAM, εφαρμόζονται τεχνικές δυναμικής αποδέσμευσης πόρων αμέσως μετά από κάθε ολοκληρωμένη μεταφορά δεδομένων, διασφαλίζοντας τη μακροχρόνια και αδιάλειπτη λειτουργία του σταθμού.

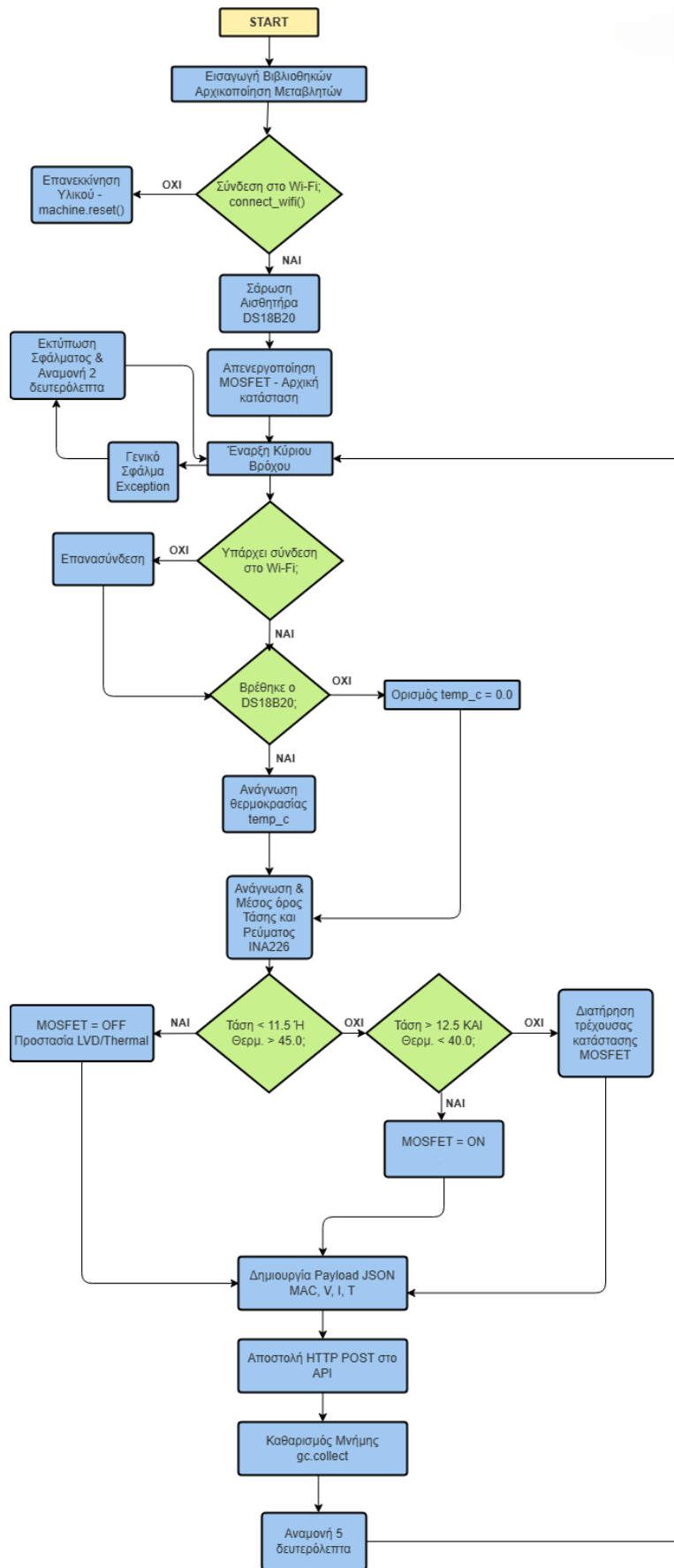
```
response = None
try:
    response = requests.post(URL, json=payload)
    print(f"Status: {response.status_code}")
except Exception as req_e:
    print(f"[HTTP ERROR] Αποτυχία αποστολής: {req_e}")
finally:
    if response:
        response.close()

gc.collect()
```

Εικόνα 7.4: Διαδικασία αποστολής HTTP POST και δυναμική εκκαθάριση μνήμης

7.5 Διάγραμμα ροής (Flow Chart) Προγράμματος ESP32

Παρακάτω στο Σχήμα 7.1 δίνεται το διάγραμμα ροής για τον προγραμματισμό του ESP32 που βασίζεται στο αρχείο main.py



Σχήμα 7.1: Διάγραμμα Ροής Προγράμματος ESP32

7.6 Εξήγηση Διαγράμματος Ροής

Το λογισμικό που εκτελείται στον μικροελεγκτή ESP32 (main.py) ακολουθεί μια αυστηρά καθορισμένη σειριακή λογική, η οποία απεικονίζεται στο Διάγραμμα Ροής της προηγούμενης ενότητας στο Σχήμα 7.1.

7.6.1 Αρχικοποίηση και Έλεγχος Δικτύου

Βλέποντας την Εικόνα 7.5, κατά την έναρξη του προγράμματος, το σύστημα εισάγει τις απαραίτητες βιβλιοθήκες και αρχικοποιεί τα πρωτόκολλα επικοινωνίας (I2C, 1-Wire), καθώς και τους ακροδέκτες (Pins) του μικροελεγκτή. Στη συνέχεια, καλείται η συνάρτηση connect_wifi(). Εδώ εφαρμόζεται ένας μηχανισμός χρονοδιακόπτη (timeout) 20 δευτερολέπτων. Εάν ο μικροελεγκτής δεν καταφέρει να λάβει διεύθυνση IP εντός αυτού του χρόνου (π.χ. λόγω απώλειας σήματος του router), εκτελείται αυτόματη επανεκκίνηση υλικού (machine.reset()) για την επαναφορά του συστήματος σε καθαρή κατάσταση.

```
import network
import urequests as requests
import time
import gc
from machine import Pin, I2C
import ustruct
import onewire
import ds18x20
import secrets

i2c = I2C(0, scl=Pin(22), sda=Pin(21))
INA226_ADDR = 0x40
R_TOTAL = 0.0014778

ds_pin = Pin(4)
ds_sensor = ds18x20.DS18X20(OneWire(ds_pin))
mosfet = Pin(15, Pin.OUT)
```

Εικόνα 7.5: Εισαγωγή απαραίτητων βιβλιοθηκών - αρχικοποίηση hardware

Στο απόσπασμα αρχικοποίησης της Εικόνας 7.5, ορίζονται οι βασικές παράμετροι διασύνδεσης του υλικού:

- Ο δίαυλος I2C δεσμεύει τις ακίδες GPIO 22 (SCL) και GPIO 21 (SDA) για την επικοινωνία με τον αισθητήρα INA226, ο οποίος είναι ρυθμισμένος στην προεπιλεγμένη διεύθυνση 0x40 (όταν οι ακροδέκτες A0 και A1 είναι γειωμένοι).
- Η σταθερά $R_TOTAL = 0.0014778$ αντιπροσωπεύει την πραγματική ισοδύναμη αντίσταση μέτρησης ρεύματος. Προκύπτει από την παράλληλη σύνδεση της ενσωματωμένης αντίστασης SMD 0.1Ω που φέρει η πλακέτα CJMCU-226 με την εξωτερική αντίσταση Shunt 50A/75mV σύμφωνα με τη σχέση 7.1.

$$R_{total} = \frac{0.1 \times 0.0015}{0.1 + 0.0015} \approx 0.0014778 \Omega \quad (7.1)$$

- Η ακίδα GPIO 4 δηλώνεται ως γραμμή δεδομένων του πρωτοκόλλου 1-Wire για τον ψηφιακό θερμοστάτη DS18B20, ενώ η ακίδα GPIO 15 ρυθμίζεται ως ψηφιακή έξοδος (Output) για την οδήγηση της πύλης του κυκλώματος διπλού MOSFET.

7.6.2 Σάρωση Αισθητήρων και Εκκίνηση

Εφόσον η σύνδεση στο δίκτυο επιτευχθεί, το σύστημα εκτελεί σάρωση στον δίαυλο 1-Wire για την ανεύρεση του αισθητήρα θερμοκρασίας DS18B20. Εάν ο αισθητήρας δεν εντοπιστεί, το πρόγραμμα δεν

τερματίζεται, αλλά θέτει μια προεπιλεγμένη τιμή ασφαλείας (`temp_c = 0.0`) για να αποφευχθούν σφάλματα υπολογισμού. Παράλληλα, το κύκλωμα ελέγχου (MOSFET) τίθεται αρχικά σε κατάσταση απενεργοποίησης (OFF) για λόγους ασφαλείας, προτού ξεκινήσει η κανονική ροή.

7.6.3 Κύριος Βρόχος και Τοπικός Έλεγχος (Edge BMS)

Η καρδιά του προγράμματος είναι ένας αέναος κύριος βρόχος που υπάρχει στην κύρια συνάρτηση `main`. Σε κάθε επανάληψη, το σύστημα αρχικά επιβεβαιώνει την ύπαρξη σύνδεσης Wi-Fi. Αμέσως μετά, εκτελείται η ανάγνωση των αισθητήρων. Για τον υπολογισμό της τάσης και του ρεύματος (INA226), καλείται η συνάρτηση `get_averaged_ina_readings()`, η οποία χρησιμοποιεί τεχνικές υπερδειγματοληψίας (oversampling 5 δειγμάτων) για την εξάλειψη του ηλεκτρικού θορύβου.

Στο σημείο αυτό εκτελείται ο αλγόριθμος τοπικού ελέγχου (Edge BMS) που φαίνεται στην Εικόνα 7.6:

- Εάν η τάση πέσει κάτω από 11.5V (κίνδυνος βαθιάς εκφόρτισης) ή η θερμοκρασία ξεπεράσει τους 45°C, το MOSFET απενεργοποιείται (OFF) για την προστασία της μπαταρίας.
- Εάν η τάση ανακάμψει πάνω από 12.5V και η θερμοκρασία είναι κάτω από 40°C, το φορτίο επανασυνδέεται (ON).

```
if real_voltage < 11.5 or temp_c > 45.0:
    mosfet.value(0)
elif real_voltage > 12.5 and temp_c < 40.0:
    mosfet.value(1)
```

Εικόνα 7.6: Αλγόριθμος τοπικού ελέγχου για την προστασία της μπαταρίας

7.6.4 Αποστολή Δεδομένων και Διαχείριση Μνήμης

Στο τελικό στάδιο του βρόχου, οι μετρήσεις και η κατάσταση του φορτίου μορφοποιούνται σε ένα αντικείμενο JSON (payload). Μέσω της εντολής `requests.post()`, το πακέτο αποστέλλεται στο Web API. Η διαδικασία αυτή περικλείεται σε μπλοκ `try...except...finally`, ώστε να κλείνει υποχρεωτικά η δικτυακή αποκριση (`response.close()`) ανεξαρτήτως έκβασης. Αμέσως μετά, καλείται ο Συλλέκτης Απορριμμάτων (`gc.collect()`) για την αποδέσμευση της μνήμης RAM, και το πρόγραμμα εισέρχεται σε αναμονή 5 δευτερολέπτων πριν ξεκινήσει τον επόμενο κύκλο μετρήσεων.

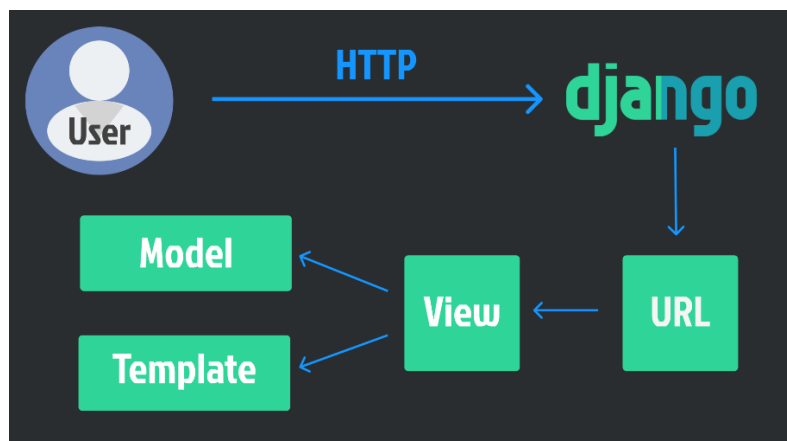
Κεφάλαιο 8ο: Επίπεδο Εφαρμογής (Application Layer)

8.1 Εισαγωγή

Στο παρόν κεφάλαιο αναλύεται το Επίπεδο Εφαρμογής (Application Layer), το οποίο αποτελεί το ανώτερο επίπεδο της αρχιτεκτονικής του συστήματος IoT. Ο ρόλος του είναι η παραλαβή, αποθήκευση, επεξεργασία και τελική παρουσίαση των τηλεμετρικών δεδομένων που αποστέλλονται από τον μικροελεγκτή. Η ανάπτυξη του Backend υλοποιήθηκε με χρήση του πλαισίου ανάπτυξης (framework) Django της Python, ενώ η μόνιμη αποθήκευση των δεδομένων βασίζεται στο σύστημα σχεσιακών βάσεων δεδομένων PostgreSQL. Το επίπεδο αυτό αναλαμβάνει το βαρύ υπολογιστικό έργο, εκτελώντας τους αλγορίθμους εκτίμησης της Κατάστασης Φόρτισης (SoC) και της Κατάστασης Υγείας (SoH) της μπαταρίας.

8.2 Αρχιτεκτονική του Django (MTV)

Το Django ακολουθεί το αρχιτεκτονικό πρότυπο MTV (Model-Template-View) που υπάρχει στην Εικόνα 8.1, το οποίο επιτυγχάνει τον σαφή διαχωρισμό της λογικής των δεδομένων από τη διεπαφή του χρήστη και τη διαχείριση των αιτημάτων. Αυτός ο διαχωρισμός προσφέρει ευελιξία, ασφάλεια και ευκολία στη συντήρηση του κώδικα.



Εικόνα 8.1: Το αρχιτεκτονικό πρότυπο MTV [21]

8.2.1 Τα Μοντέλα (Models)

Τα Μοντέλα (Models) συνιστούν την αποκλειστική πηγή αλήθειας αναφορικά με τη δομή των δεδομένων της εφαρμογής. Στο πλαίσιο αυτό, οι οντότητες του συστήματος (όπως, ενδεικτικά, οι συσκευές και οι μετρήσεις) αναπαρίστανται μέσω κλάσεων της γλώσσας προγραμματισμού Python. Κάθε κλάση-μοντέλο αντιστοιχεί σε έναν πίνακα της βάσης δεδομένων, ενώ κάθε χαρακτηριστικό (attribute) της αντιστοιχεί σε μια στήλη του εν λόγω πίνακα. Η αντιστοίχιση αυτή καθίσταται δυνατή μέσω του ενσωματωμένου μηχανισμού ORM (Object-Relational Mapper) του πλαισίου Django, ο οποίος επιτρέπει στον προγραμματιστή να αλληλεπιδρά με τη βάση δεδομένων αποκλειστικά μέσω κώδικα Python, χωρίς να απαιτείται η απευθείας σύνταξη σύνθετων ερωτημάτων SQL.

8.2.2 Τα Πρότυπα (Templates)

Τα Πρότυπα (Templates) είναι επιφορτισμένα με το επίπεδο παρουσίασης (Frontend) της εφαρμογής. Πρόκειται για αρχεία HTML στα οποία ενσωματώνεται η γλώσσα προτύπων του Django (Django

Template Language - DTL). Μέσω της DTL καθίσταται δυνατή η δυναμική έγχυση (injection) δεδομένων προερχόμενων από τη βάση (όπως χρονικές σημάνσεις και τιμές τάσης) εντός της ιστοσελίδας, τα οποία εν συνεχεία μεταβιβάζονται στη JavaScript και στη βιβλιοθήκη Chart.js με σκοπό τη δημιουργία των αντίστοιχων γραφημάτων.

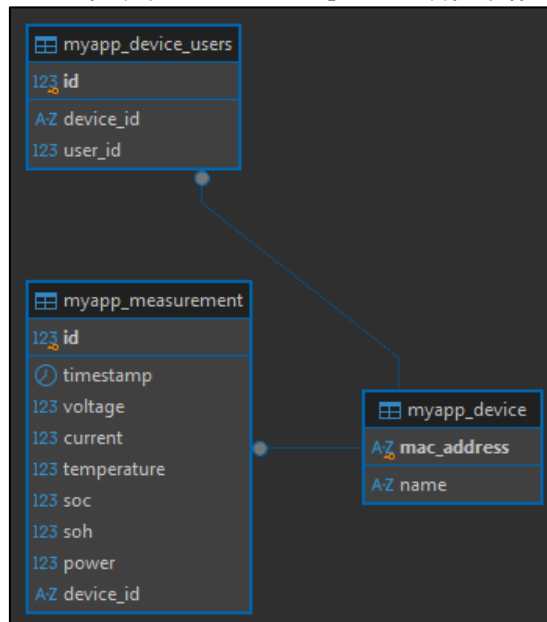
8.2.3 Οι Προβολές (Views)

Οι Προβολές (Views), από την πλευρά τους, επιτελούν τον ρόλο του συντονιστή (controller) του συστήματος. Αναλαμβάνουν την υποδοχή των αιτημάτων HTTP (Requests), την εφαρμογή της επιχειρησιακής λογικής (business logic), καθώς και την ανάκτηση ή αποθήκευση δεδομένων μέσω των Models, προκειμένου εν τέλει να επιστρέψουν την κατάλληλη απάντηση (Response). Στο πλαίσιο της παρούσας εφαρμογής αναπτύχθηκαν δύο κατηγορίες Views: αφενός οι Προβολές API, οι οποίες παραλαμβάνουν τα δεδομένα (JSON) από τη συσκευή ESP32, και αφετέρου οι Προβολές διεπαφής (UI), οι οποίες είναι υπεύθυνες για την εξυπηρέτηση των αρχείων HTML προς τους τελικούς χρήστες.

8.3 Σχεδιασμός Βάσης Δεδομένων

Λαμβάνοντας υπόψη τον σημαντικό όγκο δεδομένων που παράγεται από τον αισθητήρα, κρίθηκε αναγκαία η υιοθέτηση μιας αξιόπιστης και ισχυρής βάσης δεδομένων. Για τον λόγο αυτό επιλέχθηκε η PostgreSQL, φιλοξενούμενη στην πλατφόρμα Neon. Ο σχεδιασμός της βάσης ακολουθεί πλήρως το σχεσιακό μοντέλο και θεμελιώνεται στις εξής βασικές οντότητες:

1. **Πίνακας Συσκευών (Device):** Στον πίνακα αυτόν αποθηκεύονται τα μοναδικά αναγνωριστικά κάθε συσκευής (διεύθυνση MAC), η ονομασία της αντίστοιχης εγκατάστασης, καθώς και τα βασικά τεχνικά χαρακτηριστικά της μπαταρίας όπως η ονομαστική τάση και η χωρητικότητα.
2. **Πίνακας Μετρήσεων (Measurement):** Αποτελεί τον πυρήνα του υποσυστήματος τηλεμετρίας, καθώς σε αυτόν καταγράφονται η Τάση (V), το Ρεύμα (A), η Θερμοκρασία (°C), καθώς και το αυτόματα παραγόμενο Timestamp κάθε εγγραφής.



Εικόνα 8.2: ERD Βάσης Δεδομένων

Για την καλύτερη οπτική κατανόηση της αρχιτεκτονικής των δεδομένων, στην Εικόνα 8.2 παρατίθεται το παραπάνω διάγραμμα Οντοτήτων-Συσχετίσεων (ERD). Το σχήμα της βάσης δεδομένων εξασφαλίζει την ακεραιότητα της πληροφορίας μέσω των εξής συσχετίσεων:

- **Σχέση Συσκευής - Μετρήσεων (1:N):** Μεταξύ του πίνακα συσκευών (`myapp_device`) και του πίνακα μετρήσεων (`myapp_measurement`) ορίζεται σχέση «ένα-προς-πολλά» (One-to-Many). Το πρωτεύον κλειδί της συσκευής (`mac_address`) λειτουργεί ως ξένο κλειδί (`device_id`) στον πίνακα των μετρήσεων. Αυτό σημαίνει ότι κάθε συσκευή παράγει, στο πέρασμα του χρόνου, πλήθος χιλιάδων εγγραφών, οι οποίες αντιστοιχίζονται αυστηρά στη συγκεκριμένη εγκατάσταση.
- **Σχέση Συσκευής - Χρηστών (N:M):** Στο διάγραμμα αποτυπώνεται και ένας τρίτος, ενδιάμεσος πίνακας με την ονομασία `myapp_device_users`. Ο πίνακας αυτός υλοποιεί τη σχέση «πολλά-προς-πολλά» (Many-to-Many) μεταξύ των κόμβων IoT και των εγγεγραμμένων χρηστών του συστήματος (System Users). Περιέχοντας τα ξένα κλειδιά `device_id` και `user_id`, επιτρέπει αφενός σε έναν χρήστη (π.χ. διαχειριστή στόλου) να εποπτεύει πολλαπλές εγκαταστάσεις, και αφετέρου σε μια συσκευή να παρακολουθείται ταυτόχρονα από πολλούς διαφορετικούς χρήστες.

8.3.1 Διαχείριση Δεδομένων Χρονοσειρών (Time-Series Data) στο IoT

Τα τηλεμετρικά δεδομένα που παράγονται από τα συστήματα του Διαδικτύου των Πραγμάτων (IoT), όπως η συνεχής καταγραφή της τάσης, του ρεύματος και της θερμοκρασίας του συσσωρευτή, ανήκουν στην κατηγορία των Δεδομένων Χρονοσειρών (Time-Series Data). Ένα από τα βασικά διλήμματα κατά τον σχεδιασμό της αρχιτεκτονικής του backend είναι η επιλογή μεταξύ Σχεσιακών (SQL) και Μη-Σχεσιακών (NoSQL) βάσεων δεδομένων.

Από τη μία πλευρά, οι βάσεις NoSQL (π.χ. MongoDB, InfluxDB) προσφέρουν εξαιρετική ευελιξία και ταχύτητα εγγραφής (write performance), καθιστώντας τες δημοφιλείς σε εφαρμογές IoT τεράστιας κλίμακας, όπου τα δεδομένα δεν έχουν αυστηρή δομή. Ωστόσο, υστερούν συχνά στη διασφάλιση των ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability).

Από την άλλη πλευρά, στην παρούσα υλοποίηση επιλέχθηκε συνειδητά το Σχεσιακό Μοντέλο (SQL) και συγκεκριμένα η PostgreSQL [30], για τους εξής καθοριστικούς λόγους:

- **Αυστηρή Δομή και Ακεραιότητα:** Η εφαρμογή του Τοπικού Συστήματος Διαχείρισης Μπαταρίας απαιτεί απόλυτη συνέπεια. Κάθε μέτρηση πρέπει να συσχετίζεται αυστηρά με έναν μοναδικό κόμβο (Device MAC Address) και με τους αντίστοιχους χρήστες που έχουν δικαίωμα προβολής.
- **Σταθερότητα Σχήματος (Schema):** Το παραγόμενο ωφέλιμο φορτίο (JSON payload) του μικροελεγκτή ESP32 είναι αυστηρά καθορισμένο, χωρίς μεταβαλλόμενα πεδία, συνεπώς η "ευελιξία" αδόμητων δεδομένων των NoSQL δεν προσφέρει πρακτικό πλεονέκτημα.
- **Ισχυρά Ερωτήματα Ανάλυσης:** Η PostgreSQL επιτρέπει την εκτέλεση εξαιρετικά σύνθετων αναλυτικών ερωτημάτων (SQL Queries) στο ιστορικό του συσσωρευτή, τα οποία είναι απαραίτητα για την εκτίμηση του State of Health (SoH) μέσω συγκρίσεων παλαιότερων τιμών εσωτερικής αντίστασης.

Επιπλέον, σύμφωνα με πρόσφατες συγκριτικές μελέτες, η PostgreSQL αποδεικνύεται εξαιρετικά αποδοτική σε περιβάλλοντα που απαιτούν χαμηλή καθυστέρηση δεδομένων και ταυτόχρονη εκτέλεση λειτουργιών [29]. Διατηρεί κορυφαία σταθερότητα ακόμη και κατά την παράλληλη εκτέλεση ερωτημάτων ανάγνωσης και συνεχούς εισαγωγής νέων εγγραφών [29], χαρακτηριστικό απολύτως απαραίτητο για ένα σύστημα IoT όπου η τηλεμετρία εισέρχεται αδιάλειπτα την ίδια στιγμή που ο χρήστης ανακτά το ιστορικό στο Dashboard.

8.3.2 Serverless Αρχιτεκτονική και Υπολογιστικό Νέφος (Neon PostgreSQL)

Μια επιπλέον καινοτομία της αρχιτεκτονικής του συστήματος είναι η αξιοποίηση της τεχνολογίας Serverless Cloud Computing για τη φιλοξενία της βάσης δεδομένων, με τη χρήση της πλατφόρμας Neon.

Ιστορικά, η φιλοξενία μιας βάσης δεδομένων στο υπολογιστικό νέφος απαιτούσε την εκμίσθωση μιας εικονικής μηχανής (Virtual Machine) ή ενός Managed Database Instance (π.χ. AWS RDS), τα οποία εκτελούνται αδιάλειπτα 24 ώρες το 24ωρο. Αυτό το μοντέλο, αν και αξιόπιστο, συνεπάγεται συνεχή κατανάλωση υπολογιστικών πόρων και συνεπώς σταθερό λειτουργικό κόστος, ανεξάρτητα από τον πραγματικό φόρτο εργασίας.

Η Serverless αρχιτεκτονική επιλύει αυτό το πρόβλημα εφαρμόζοντας τις εξής αρχές:

- **Διαχωρισμός Αποθήκευσης και Επεξεργασίας (Separation of Storage and Compute):** Τα δεδομένα των μετρήσεων αποθηκεύονται σε κατανεμημένο, φθινό αποθηκευτικό χώρο (cloud storage), ανεξάρτητα από τους επεξεργαστές που εκτελούν τα ερωτήματα, κάτι που αποτελεί τον θεμέλιο λίθο των σύγχρονων Cloud-Native συστημάτων [32], [33].
- **Αυτόματη Κλιμάκωση (Auto-scaling & Scale-to-Zero):** Όταν δεν υπάρχουν εισερχόμενα τηλεμετρικά δεδομένα από τα φωτοβολταϊκά και κανένας διαχειριστής δεν εξετάζει το Dashboard, η πλατφόρμα μηδενίζει (scale-to-zero) τους υπολογιστικούς πόρους [31]. Όταν αποστέλλεται ένα νέο HTTP POST αίτημα από τον μικροελεγκτή, η επεξεργαστική ισχύς επανέρχεται σε κλάσματα του δευτερολέπτου.
- **Βελτιστοποίηση Κόστους για Στόλους IoT:** Σε εφαρμογές Fleet Management αυτόνομων συστημάτων, η ροή δεδομένων εμφανίζει συχνά διακυμάνσεις. Το Serverless μοντέλο (pay-as-you-go) εγγυάται ότι ο πάροχος της υπηρεσίας χρεώνει αποκλειστικά για τα χιλιοστά του δευτερολέπτου κατά τα οποία υπήρξε ενεργή εκτέλεση κώδικα, καθιστώντας το σύστημα οικονομικά αποδοτικό και ευέλικτο.

8.4 Λογική Backend

Η αφαίρεση του υπολογιστικού φόρτου από τον μικροελεγκτή και η μεταφορά του στον διακομιστή (Cloud/Backend) επιτρέπει την εκτέλεση προηγμένων αλγορίθμων δίχως περιορισμούς στη μνήμη ή στην επεξεργαστική ισχύ.

8.4.1 Ο Αλγόριθμος Υπολογισμού SoC (Python Backend)

Η Κατάσταση Φόρτισης (SoC - State of Charge) αποτελεί το πιο κρίσιμο μέγεθος για τη διαχείριση της ενέργειας, υποδεικνύοντας το ποσοστό της διαθέσιμης ενέργειας (0-100%). Ο υπολογισμός στον διακομιστή βασίζεται στη μέθοδο της Τάσης Ανοχτού Κυκλώματος (OCV - Open Circuit Voltage).

Όταν τα δεδομένα φτάνουν στο Backend, ο αλγόριθμος σε Python αντιστοιχίζει την τρέχουσα τάση με χαρακτηριστικές καμπύλες εκφόρτισης των μπαταριών VRLA. Για παράδειγμα, μια τάση της τάξης των 12.7V (σε ηρεμία) υποδηλώνει SoC 100%, ενώ τάση 11.8V υποδηλώνει SoC κοντά στο 25%. Ο αλγόριθμος χρησιμοποιεί τη θερμοκρασία (από τον αισθητήρα DS18B20) για να διορθώσει μικρές αποκλίσεις, καθώς η ηλεκτροχημική απόδοση της μπαταρίας μεταβάλλεται σε ακραίες θερμοκρασίες.

Πίνακας 8.1: Υπολογισμός του SoC

```
if current_voltage >= 12.7:
    new_SoC = 100.0
elif current_voltage <= 11.5:
    new_SoC = 0.0
else:
    new_SoC = ((current_voltage - 11.5) / 1.2) * 100.0
```

```

if last_measurement and last_measurement.Soc is not None:
    time_diff_seconds = (now - last_measurement.timestamp).total_seconds()
    time_diff_hours = time_diff_seconds / 3600.0
    battery_capacity = getattr(device, 'capacity_ah', BATTERY_CAPACITY_AH)
    delta_SoC = ((current_amps * time_diff_hours) / battery_capacity) * 100
    new_SoC = last_measurement.Soc + delta_SoC
    new_SoC = max(0.0, min(100.0, new_SoC)) # Περιορισμός στο [0, 100]

```

Ο αλγόριθμος του Πίνακα 8.1 λειτουργεί σε δύο διαδοχικές φάσεις:

1. Αρχική Εκτίμηση βάσει Τάσης (OCV Approximation): Εάν το σύστημα ξεκινά τη λειτουργία του χωρίς προηγούμενο ιστορικό μετρήσεων, εφαρμόζει μια γραμμική αντιστοίχιση της τάσης ηρεμίας. Τιμές τάσης $\geq 12.7V$ αποδίδουν άμεσα SoC 100%, τιμές $\leq 11.5V$ αντιστοιχούν σε SoC 0%, ενώ στο ενδιάμεσο εύρος των 1.2V το ποσοστό προκύπτει αναλογικά.
2. Δυναμικός Υπολογισμός μέσω Coulomb Counting: Εφόσον εντοπιστεί προγενέστερη καταχώρηση στη βάση δεδομένων (if last_measurement), ο αλγόριθμος υπολογίζει την ακριβή χρονική διαφορά (Δt) μεταξύ της τρέχουσας και της προηγούμενης μέτρησης σε ώρες (time_diff_hours). Ακολουθώντας, υπολογίζει τη μεταβολή του φορτίου (delta_SoC) διαιρώντας το γινόμενο του διερχόμενου ρεύματος επί τον χρόνο με την ονομαστική χωρητικότητα της μπαταρίας (battery_capacity = 4.0Ah). Η μεταβολή αυτή προστίθεται στο προηγούμενο SoC, επιτρέποντας την καταγραφή τόσο της φόρτισης (θετικό ρεύμα) όσο και της εκφόρτισης (αρνητικό ρεύμα). Τέλος, μέσω της συνάρτησης $\max(0.0, \min(100.0, \text{new_SoC}))$, η τιμή περιορίζεται αυστηρά εντός του φυσικού εύρους [0%,100%].

8.4.2 Ο Αλγόριθμος Αξιολόγησης SoH

Η Κατάσταση Υγείας (SoH - State of Health) αποτελεί δείκτη μακροπρόθεσμης φθοράς της μπαταρίας, συγκρίνοντας την τρέχουσα ικανότητά της με αυτήν μιας ολοκαίνουργιας. Στο Backend, το σύστημα παρακολουθεί δύο βασικές παραμέτρους για την εξαγωγή αυτού του δείκτη:

1. **Βαθιές Εκφορτίσεις:** Το Django καταγράφει πόσες φορές η τάση της μπαταρίας έπεσε κάτω από τα όρια ασφαλείας πριν προλάβει να επέμβει το τοπικό σύστημα διαχείρισης (Edge BMS).
2. **Πτώση Τάσης υπό Φορτίο (Voltage Sag):** Συγκρίνοντας την τάση αμέσως πριν την ενεργοποίηση του φορτίου (Λαμπτήρας P21W) και αμέσως μετά (κατά τη ροή του ρεύματος), ο αλγόριθμος εξάγει συμπεράσματα για την εσωτερική αντίσταση της μπαταρίας. Μια υγιής μπαταρία παρουσιάζει ελάχιστη πτώση, ενώ μια γερασμένη μπαταρία καταγράφει απότομη βύθιση της τάσης.

Πίνακας 8.2: Υπολογισμός του SoH

```

new_SoH = last_measurement.SoH if last_measurement.SoH else 100.0
if current_voltage < 11.5:
    new_SoH -= 0.005
if current_temp > 35.0:
    new_SoH -= 0.001 * (current_temp - 35.0)
if abs(current_amps) > 1.0:
    new_SoH -= 0.001

```

$$\text{new_SoH} = \max(0.0, \min(100.0, \text{new_SoH}))$$

Στον Πίνακα 8.2 παρουσιάζεται ο εμπειρικός αλγόριθμος υποβάθμισης της Κατάστασης Υγείας (SoH) που εκτελείται στο backend:

- Κατά την εκτέλεση, ο αλγόριθμος ανακτά την τελευταία καταγεγραμμένη τιμή SoH της συσκευής, ή θέτει την προεπιλεγμένη τιμή 100% εάν πρόκειται για καινούργια εγκατάσταση.
- Ποινή Βαθιάς Εκφόρτισης: Εάν η τάση πέσει κάτω από το κρίσιμο όριο ασφαλείας των 11.5V, αφαιρείται ποινή 0.005% ανά κύκλο μέτρησης, προσομοιώνοντας τη συσσώρευση κρυστάλλων θεικού μολύβδου στις πλάκες.
- Θερμική Καταπόνηση: Εάν η θερμοκρασία του συσσωρευτή ξεπεράσει τους 35°C, επιβάλλεται κλιμακωτή ποινή υποβάθμισης ανάλογη της υπέρβασης, σύμφωνα με τον τύπο $0.001 \times (T - 35.0)\%$, αντανakλώντας την επιτάχυνση της διάβρωσης.
- Καταπόνηση Υψηλού Ρεύματος: Ρεύματα εκφόρτισης ή φόρτισης άνω του 1.0A (>0.25C για μπαταρία 4Ah) επιβαρύνουν επιπλέον τον δείκτη υγείας κατά 0.001%.
- Η τελική τιμή του SoH περιορίζεται στο διάστημα [0%,100%] και αποθηκεύεται στον αντίστοιχο πίνακα μετρήσεων της βάσης PostgreSQL.

8.4.3 Υπολογισμός Βάθους Εκφόρτισης (DoD) και Εσωτερικής Αντίστασης (IR)

Πέραν των βασικών δεικτών, το σύστημα επεξεργάζεται τα τηλεμετρικά δεδομένα για την εξαγωγή επιπλέον δεικτών υγείας. Το Βάθος Εκφόρτισης (Depth of Discharge - DoD) υπολογίζεται άμεσα ως το συμπληρωματικό μέγεθος της Κατάστασης Φόρτισης, χρησιμοποιώντας τη σχέση (8.1).

$$DoD = 100 - SoC \quad (8.1)$$

Η συγκεκριμένη μετρική είναι κρίσιμη για την παρακολούθηση της καταπόνησης της μπαταρίας.

Επιπρόσθετα, αναπτύχθηκε αλγόριθμος εκτίμησης της Εσωτερικής Αντίστασης (Internal Resistance - IR) σε πραγματικό χρόνο. Ο αλγόριθμος συγκρίνει συνεχώς την τρέχουσα με την προηγούμενη μέτρηση τάσης και ρεύματος. Όταν ανιχνευθεί απότομη μεταβολή του φορτίου (διαφορά ρεύματος άνω των 0.5A), το σύστημα υπολογίζει την εσωτερική αντίσταση εφαρμόζοντας τον νόμο του Ohm, όπως αναφέρεται στη σχέση (8.2).

$$R_{in} = \frac{\Delta V}{\Delta I} \quad (8.2)$$

Η εξαγωγή αυτού του δείκτη επιτρέπει τον έγκαιρο εντοπισμό της γήρανσης των στοιχείων μολύβδου, καθώς η εσωτερική αντίσταση αυξάνεται δραματικά προς το τέλος του κύκλου ζωής της μπαταρίας.

Κατά τον υπολογισμό της εσωτερικής αντίστασης, εφαρμόζεται λογισμικό φίλτρο εύρους 10 mΩ – 1500 mΩ. Η τυπική ονομαστική τιμή μιας καινούργιας μπαταρίας VRLA 4Ah κυμαίνεται περί τα 35 – 45 mΩ, ενώ τιμές άνω των 200 – 300 mΩ υποδεικνύουν προχωρημένη γήρανση, σουλφάτωση ή κατάσταση βαθιάς εκφόρτισης (Voltage Sag).

8.5 Ταυτοποίηση και Ασφάλεια

Η ασφάλεια του επιπέδου εφαρμογής υλοποιήθηκε σε πολλαπλά επίπεδα, προστατεύοντας τόσο τα δεδομένα των αισθητήρων όσο και την πρόσβαση των χρηστών.

- **Προστασία API:** Το τελικό σημείο (endpoint) που δέχεται τα τηλεμετρικά δεδομένα δεν τα αποθηκεύει τυφλά. Απαιτείται η έγκυρη ταυτοποίηση της Συσκευής (Device Auth) μέσω της διεύθυνσης MAC. Εάν η διεύθυνση MAC δεν είναι ήδη καταχωρημένη στη βάση δεδομένων, το αίτημα απορρίπτεται άμεσα.
- **Διαχείριση Περιβάλλοντος (Environment Variables):** Κατά την εκτέλεση της εφαρμογής στο τοπικό περιβάλλον, ευαίσθητα δεδομένα όπως οι κωδικοί της PostgreSQL και το

SECRET_KEY του Django, δεν περιλαμβάνονται στον πηγαίο κώδικα. Αντίθετα, διαχειρίζονται μέσω μεταβλητών περιβάλλοντος (environment variables), ώστε να μην περιλαμβάνονται στον πηγαίο κώδικα.

- **Ταυτοποίηση Χρηστών:** Η πρόσβαση στα γραφήματα και στον πίνακα ελέγχου (Dashboard) προστατεύεται από το ενσωματωμένο σύστημα αυθεντικοποίησης του Django (Session-based Authentication). Ο χρήστης υποχρεούται να εισάγει έγκυρα διαπιστευτήρια (Username / Password) για να παρακολουθήσει τον στόλο των εγκαταστάσεών του, εξασφαλίζοντας την ιδιωτικότητα των δεδομένων

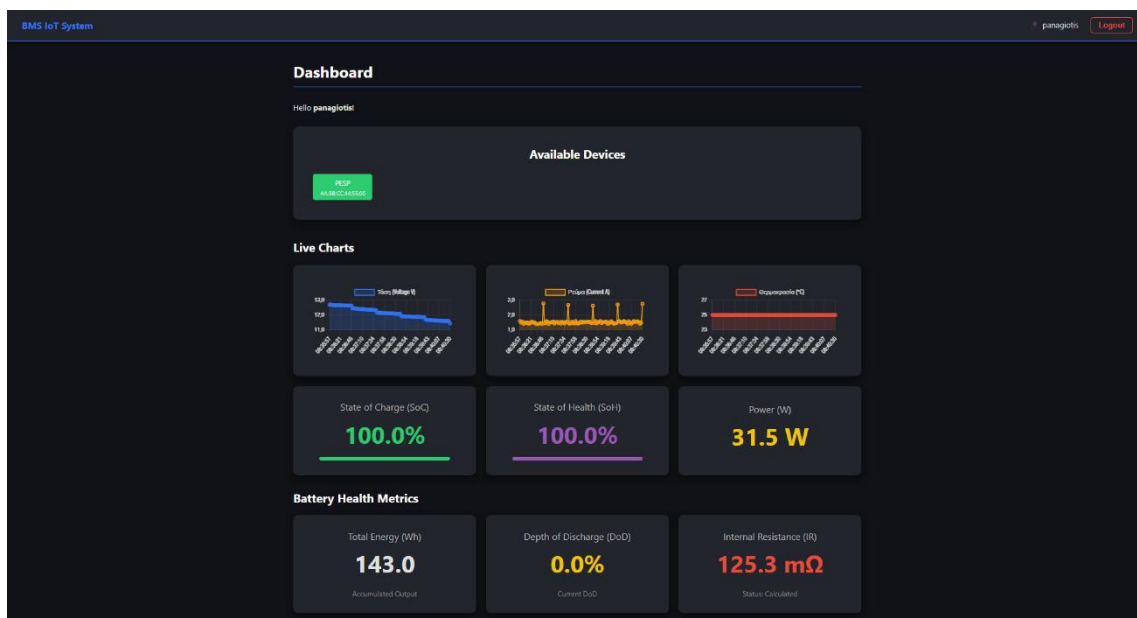
Κεφάλαιο 9ο: Διεπαφή Χρήστη (UI) και Fleet Management

9.1 Εισαγωγή

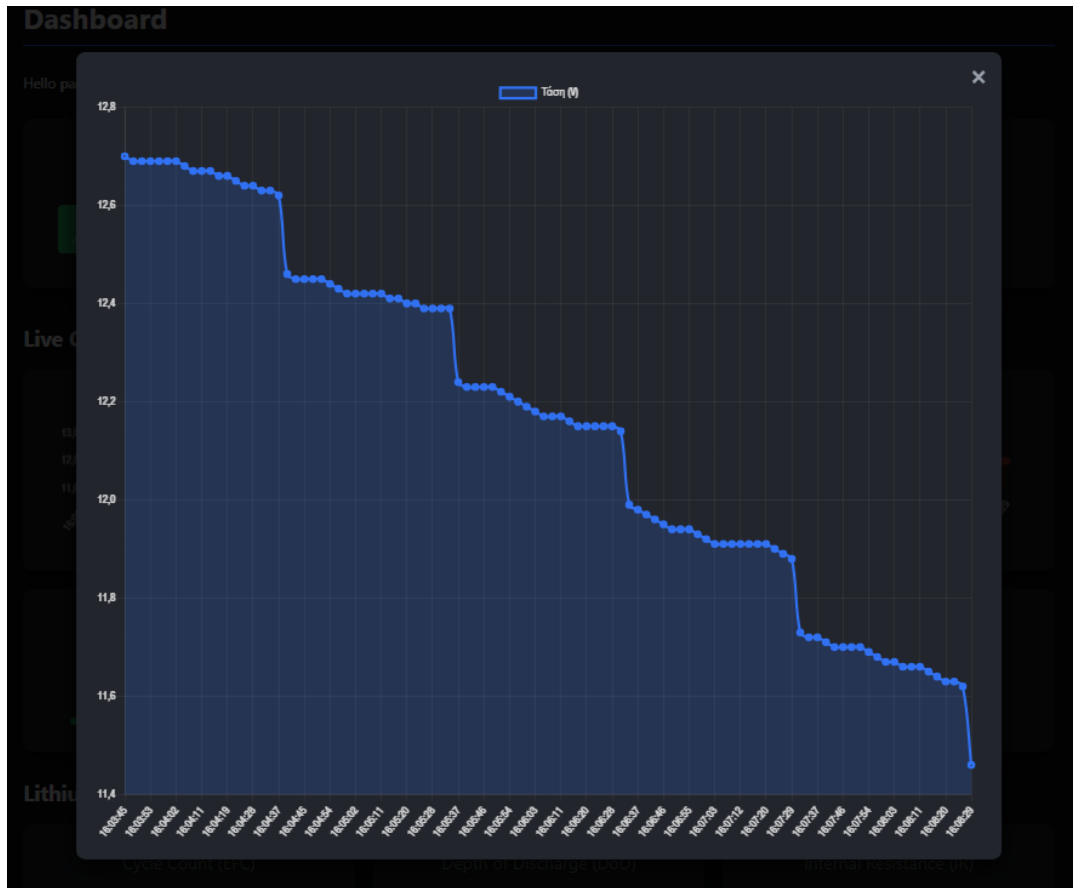
Στο παρόν κεφάλαιο παρουσιάζεται η Γραφική Διεπαφή Χρήστη (Graphical User Interface - GUI), η οποία συνιστά το τελικό σημείο επαφής μεταξύ του χρήστη και του συστήματος IoT. Κατά τον σχεδιασμό της δόθηκε ιδιαίτερη βαρύτητα στη χρηστικότητα, στην ταχύτητα απόκρισης, καθώς και στην ευχερή κατανόηση των παρουσιαζόμενων δεδομένων. Προκειμένου να καλυφθούν οι διαφορετικές ανάγκες της εφαρμογής, το σύστημα σχεδιάστηκε ώστε να υποστηρίζει δύο διακριτές οπτικές: αφενός μία εστιασμένη στον τελικό χρήστη/ιδιοκτήτη μιας μεμονωμένης εγκατάστασης (Client Dashboard) και αφετέρου μία προσανατολισμένη προς τον κεντρικό διαχειριστή, ο οποίος εποπτεύει το σύνολο των κόμβων του συστήματος (Fleet Dashboard).

9.2 Πίνακας Ελέγχου Πελάτη (Client Dashboard)

Στην κεντρική οθόνη προβάλλονται οι δείκτες απόδοσης (KPIs) όπως φαίνεται στην Εικόνα 9.1, που απεικονίζουν σε πραγματικό χρόνο τις κρίσιμότερες μετρικές: την τρέχουσα τάση, το ρεύμα, τη θερμοκρασία και την Κατάσταση Φόρτισης (SoC). Επιπλέον, προστέθηκε μια διακριτή ενότητα οπτικοποίησης προηγμένων δεικτών υγείας, η οποία περιλαμβάνει τη Συνολική Παραγόμενη Ενέργεια (Total Energy σε Wh), το Βάθος Εκφόρτισης (DoD) και τη δυναμικά υπολογιζόμενη Εσωτερική Αντίσταση (IR). Όπως φαίνεται στην Εικόνα 9.2, κάθε κάρτα ενσωματώνει λειτουργικότητα αναδεδουλευμένου παραθύρου, επιτρέποντας στον χρήστη να κάνει κλικ και να αναλύσει το αντίστοιχο ιστορικό γράφημα. Ο σχεδιασμός της διεπαφής έχει ως στόχο να παρέχει στον χρήστη τη δυνατότητα άμεσης αξιολόγησης (glanceability) της ενεργειακής του αυτονομίας, χωρίς να απαιτούνται εξειδικευμένες τεχνικές γνώσεις.



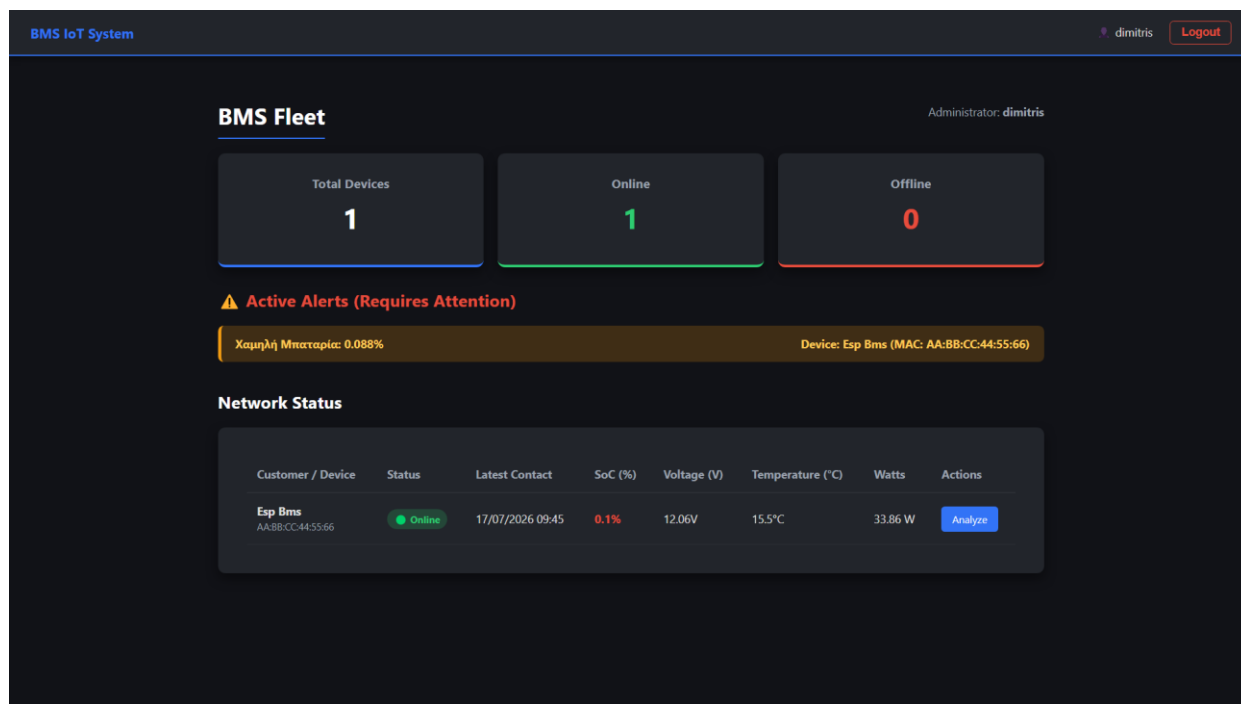
Εικόνα 9.1: Ο πίνακας ελέγχου του πελάτη



Εικόνα 9.2: Το αναδυόμενο παράθυρο για κάθε δείκτη

9.3 Κέντρο Διαχείρισης Στόλου (Fleet Dashboard)

Σε αντιδιαστολή προς τον Πίνακα Ελέγχου Πελάτη, το Κέντρο Διαχείρισης Στόλου (Fleet Management) απευθύνεται στο επίπεδο του διαχειριστή (Administrator) του συστήματος και φαίνεται στην Εικόνα 9.3. Η συγκεκριμένη προβολή παρέχει μια συνολική, μακροσκοπική εικόνα, καθώς συγκεντρώνει σε ενιαία οθόνη τα δεδομένα του συνόλου των εγγεγραμμένων κόμβων IoT (μπαταριών/συσκευών). Μέσω αυτής, ο διαχειριστής έχει τη δυνατότητα να ταξινομεί τις συσκευές με βάση την κατάσταση υγείας τους (State of Health - SoH), να εντοπίζει άμεσα τους κόμβους που βρίσκονται εκτός σύνδεσης (offline), καθώς και να παρακολουθεί συγκεντρωτικά στατιστικά στοιχεία για το σύνολο του δικτύου. Η συγκεκριμένη υλοποίηση καθιστά το σύστημα ιδιαίτερα κατάλληλο για εταιρείες συντήρησης ή παρόχους ενέργειας, οι οποίοι διαχειρίζονται πλήθος απομακρυσμένων φωτοβολταϊκών εγκαταστάσεων.



Εικόνα 9.3: Ο πίνακας ελέγχου του διαχειριστή

9.4 Σύστημα Συναγερμών (Alerts)

Η παρακολούθηση σε πραγματικό χρόνο συμπληρώνεται από ένα ενεργητικό και αυτοματοποιημένο Σύστημα Συναγερμών (Alerts). Το σύστημα αυτό λειτουργεί στο παρασκήνιο (backend) του Django και ελέγχει τα εισερχόμενα τηλεμετρικά δεδομένα έναντι προκαθορισμένων ορίων ασφαλείας. Εκτός από τις οπτικές ενδείξεις στη Διεπαφή Χρήστη, υλοποιήθηκε μηχανισμός άμεσης ειδοποίησης μέσω ηλεκτρονικού ταχυδρομείου (Email Notifications), διασφαλίζοντας ότι ο διαχειριστής θα ενημερωθεί εγκαίρως ακόμη και αν δεν παρακολουθεί ενεργά την εφαρμογή.

Ο αλγόριθμος αποστολής των αυτοματοποιημένων μηνυμάτων ενεργοποιείται σε τρεις κρίσιμες περιπτώσεις λειτουργίας:

1. **Υπερθέρμανση Μπαταρίας (DANGER - Overheating):** Εάν η θερμοκρασία ξεπεράσει τους 45.0°C, αποστέλλεται σήμα άμεσου κινδύνου, καθώς η υψηλή θερμοκρασία επηρεάζει δραματικά την ηλεκτροχημική ισορροπία της μπαταρίας.
2. **Βαθιά Εκφόρτιση (DANGER - Deep Discharge):** Εάν η τάση πέσει κάτω από τα 11.5V, το σύστημα προειδοποιεί για κίνδυνο μη αναστρέψιμης καταστροφής (σουλφάτωση) των στοιχείων μολύβδου.
3. **Χαμηλό Απόθεμα Ενέργειας (WARNING - Low Battery):** Εάν η εκτιμώμενη Κατάσταση Φόρτισης (SoC) υποχωρήσει κάτω από το 20%, αποστέλλεται προειδοποιητικό μήνυμα, υπενθυμίζοντας ότι το φορτίο πρέπει να παραμείνει απενεργοποιημένο.

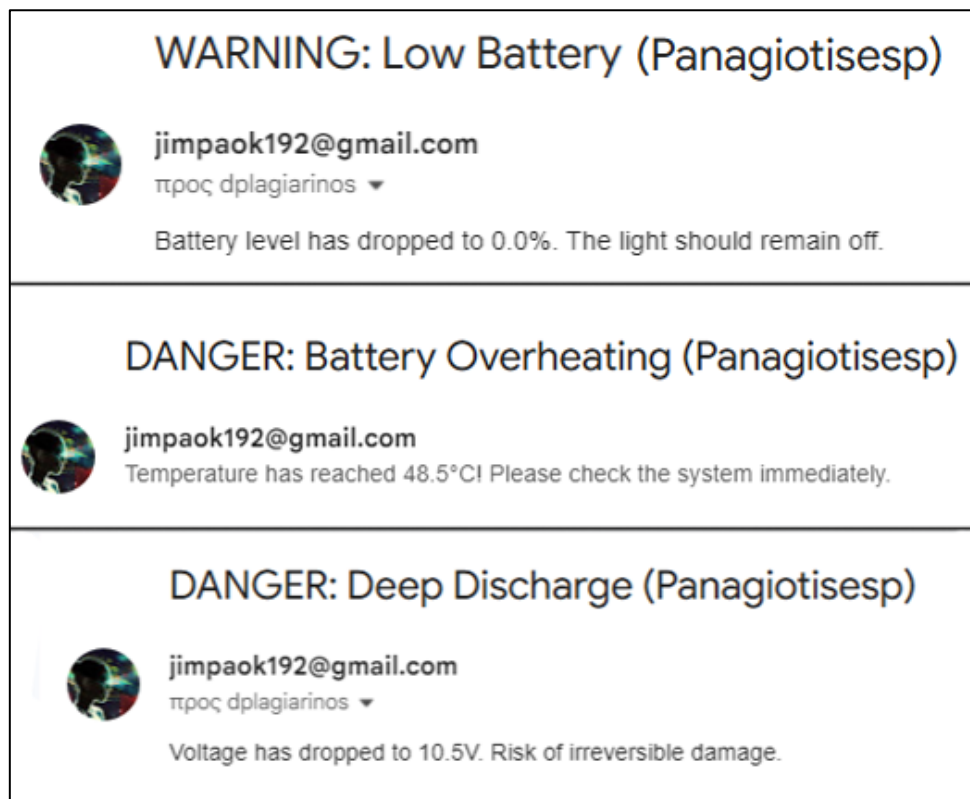
Ένα συχνό πρόβλημα στα συστήματα IoT είναι η συνεχής και ενοχλητική αποστολή ειδοποιήσεων (spamming) όταν μια τιμή παραμένει οριακά εκτός ορίων σε συνεχόμενες μετρήσεις. Για την επίλυση αυτού του προβλήματος, αναπτύχθηκε ένας μηχανισμός χρονικής υστέρησης (Cooldown Mechanism). Όπως φαίνεται και στο παρακάτω απόσπασμα κώδικα στον Πίνακα 9.1, το σύστημα αξιοποιεί την προσωρινή μνήμη (Cache) του διακομιστή δημιουργώντας ένα μοναδικό κλειδί ταυτοποίησης (cache_key) βασισμένο στη διεύθυνση MAC της εκάστοτε συσκευής. Μόλις σταλεί το πρώτο email, το

κλειδί αυτό ενεργοποιείται, "παγώνοντας" την αποστολή νέων ειδοποιήσεων για την ίδια συσκευή μέχρι να παρέλθει ένα καθορισμένο χρονικό διάστημα.

Πίνακας 9.1: Ο αλγόριθμος ελέγχου ορίων και η λογική του cooldown

```
if current_temp > 45.0:
    alert_subject = f"DANGER: Battery Overheating ({device.name})"
    alert_message = f"Temperature has reached {current_temp}°C! Please check the system immediately."
elif current_voltage < 11.5:
    alert_subject = f"DANGER: Deep Discharge ({device.name})"
    alert_message = f"Voltage has dropped to {current_voltage} V. Risk of irreversible damage."
elif new_SoC < 20.0:
    alert_subject = f"WARNING: Low Battery ({device.name})"
    alert_message = f"Battery level has dropped to {new_SoC}%. The light should remain off."
if alert_message:
    cache_key = f"alert_cooldown_{device.mac_address}"
```

Παρακάτω στην Εικόνα 9.4 φαίνονται οι 3 περιπτώσεις αποστολής email:



Εικόνα 9.4: Τα e-mail προειδοποίησης

9.5 Διαδραστικότητα και Εμπειρία Χρήστη

Ο απλός παρατεταγμένος πίνακας αριθμών δεν επαρκεί για την κατανόηση της συμπεριφοράς μιας μπαταρίας στον χρόνο. Για τον λόγο αυτό, ενσωματώθηκε η βιβλιοθήκη Chart.js, η οποία μετατρέπει τις χρονοσειρές δεδομένων (time-series data) σε δυναμικά, διαδραστικά γραφήματα.

Ο χρήστης μπορεί να παρατηρήσει την καμπύλη φόρτισης, να απομονώσει συγκεκριμένα χρονικά διαστήματα (π.χ. πτώση τάσης κατά τη διάρκεια της νύχτας) και να αναλύσει την απόδοση του συστήματος. Παράλληλα, η σελίδα σχεδιάστηκε με τις αρχές της Ανταποκρινόμενης Σχεδίασης (Responsive Design), κάνοντας χρήση σύγχρονων τεχνικών CSS. Αυτό εξασφαλίζει ότι το ταμπλό προσαρμόζεται αυτόματα και προβάλλεται άψογα σε οποιαδήποτε οθόνη, βελτιστοποιώντας την τελική εμπειρία του χρήστη.

Κεφάλαιο 10ο: Πειραματικές Δοκιμές και Αποτελέσματα

10.1 Μεθοδολογία Δοκιμών

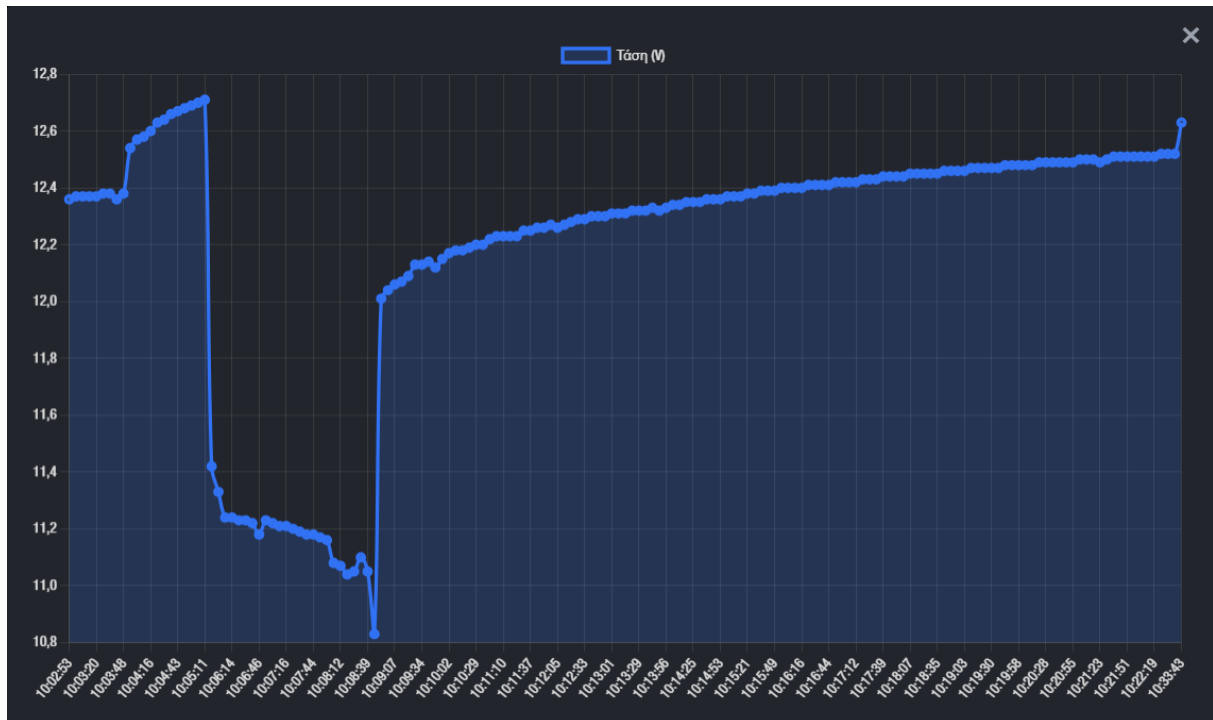
Σκοπός των πειραματικών δοκιμών ήταν η επαλήθευση της ορθής λειτουργίας του συνολικού συστήματος IoT σε πραγματικές συνθήκες, ελέγχοντας την αξιοπιστία τόσο του υλικού (Hardware) στο Επίπεδο Αντίληψης, όσο και του λογισμικού (Software) στο Επίπεδο Εφαρμογής. Η διαδικασία σχεδιάστηκε με γνώμονα την αξιολόγηση της ακρίβειας των μετρήσεων, της ταχύτητας απόκρισης του ενσωματωμένου Τοπικού Συστήματος Διαχείρισης (Edge BMS) και της ορθής καταγραφής των δεδομένων στη διαδικτυακή πλατφόρμα.

Για τη διεξαγωγή των πειραμάτων χρησιμοποιήθηκε η πλήρης φυσική διάταξη, αποτελούμενη από τη μπαταρία VRLA 12V (4Ah), το φωτοβολταϊκό πάνελ 10W και τον μικροελεγκτή ESP32. Προκειμένου να προσομοιωθεί η κατανάλωση ενέργειας μιας πραγματικής εγκατάστασης και να αξιολογηθεί η συμπεριφορά του συστήματος κάτω από πίεση (Stress Testing), αξιοποιήθηκε ως τεχνητό φορτίο ο λαμπτήρας πυρακτώσεως P21W, ο οποίος παρουσιάζει ονομαστική κατανάλωση 21 W στα 12 V, που αντιστοιχεί σε ονομαστικό ρεύμα περίπου 1,75 A.

Δεδομένου ότι ο φυσιολογικός κύκλος φόρτισης και εκφόρτισης μιας μπαταρίας απαιτεί πολλές ώρες, υιοθετήθηκε η μέθοδος της Επιταχυνόμενης Δοκιμής (Accelerated Testing). Πιο συγκεκριμένα, το σύστημα εκτέθηκε αρχικά σε συνθήκες ηλιοφάνειας ώστε το φωτοβολταϊκό πάνελ να επαναφέρει τη μπαταρία στο μέγιστο επίπεδο τάσης ηρεμίας. Στη συνέχεια, το φορτίο ενεργοποιήθηκε προκειμένου να προκαλέσει ταχεία απορρόφηση της αποθηκευμένης ενέργειας. Η πρακτική αυτή επέτρεψε την άμεση παρατήρηση του φαινομένου της πτώσης τάσης υπό φορτίο (Voltage Sag) και τη γρήγορη προσέγγιση των προκαθορισμένων κρίσιμων κατωφλίων τάσης (11.5V). Με αυτόν τον τρόπο, κατέστη δυνατή η αξιολόγηση των μηχανισμών προστασίας, δηλαδή της φυσικής αποκοπής του ρελέ (MOSFET) και του συστήματος αποστολής ειδοποιήσεων (Alerts), μέσα σε ένα αυστηρά ελεγχόμενο και σύντομο χρονικό πλαίσιο.

10.2 Αποτελέσματα Κανονικής Λειτουργίας

Κατά την πειραματική διαδικασία, το σύστημα αφέθηκε να φορτίσει τον συσσωρευτή μέσω του φωτοβολταϊκού πάνελ, φτάνοντας την τάση ηρεμίας στα 12.7V (SoC 100%). Με την ενεργοποίηση του φορτίου (λαμπτήρας 21W), καταγράφηκε ακαριαία πτώση τάσης (Voltage Sag) στα 11.5V, επιβεβαιώνοντας την ύπαρξη αυξημένης εσωτερικής αντίστασης. Αξιοσημείωτο είναι το γεγονός ότι η τάση κατέρρευσε στα 10.83V μέσα σε μόλις 5 λεπτά λειτουργίας, παρέχοντας ένδειξη αυξημένης εσωτερικής αντίστασης και πιθανής υποβάθμισης της κατάστασης του συσσωρευτή.



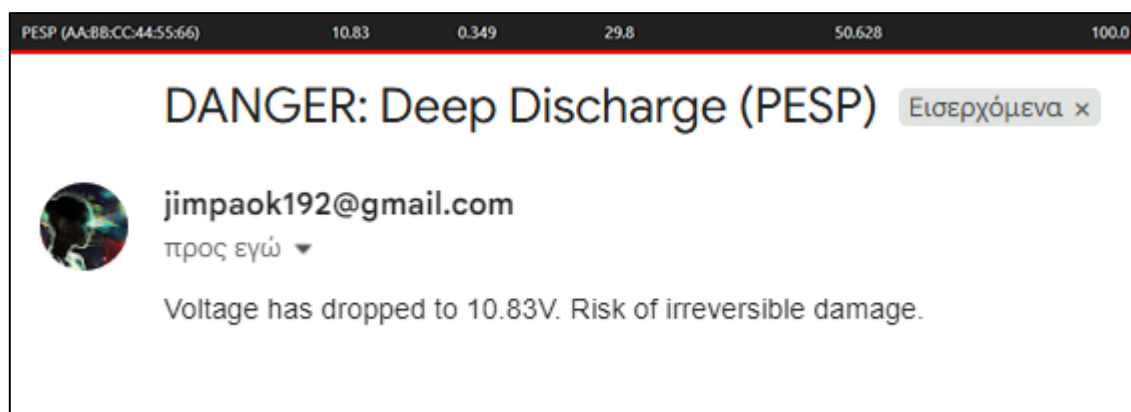
Εικόνα 10.1: Η πτώση της τάσης

10.3 Δοκιμή Συστήματος Προστασίας (LVD)

Η ραγδαία πτώση της τάσης κατά την εκφόρτιση αποτέλεσε το ιδανικό σενάριο για τη δοκιμή του Τοπικού Συστήματος Διαχείρισης (Edge BMS). Μόλις ο μικροελεγκτής ESP32 ανίχνευσε ότι η τάση έπεσε κάτω από τα κρίσιμα όρια, έδωσε εντολή απενεργοποίησης στο κύκλωμα διπλού MOSFET. Η αποκοπή του φορτίου υπήρξε ακαριαία, με αποτέλεσμα η τάση των ακροδεκτών της μπαταρίας να ανακάμψει άμεσα στα 12.0V, προστατεύοντας τα στοιχεία μολύβδου από μόνιμη βλάβη λόγω βαθιάς εκφόρτισης.

10.4 Δοκιμή Συναγερμών (Alerts)

Η καταγραφή της τάσης στα 10.83V (κάτω από το όριο των 11.5V) πυροδότησε με επιτυχία τον μηχανισμό ειδοποιήσεων του Django Backend. Όπως φαίνεται στην παρακάτω εικόνα, το σύστημα απέστειλε άμεσα το αυτοματοποιημένο μήνυμα ηλεκτρονικού ταχυδρομείου, με την ένδειξη «DANGER: Deep Discharge», ενημερώνοντας τον διαχειριστή για τον κίνδυνο μη αναστρέψιμης καταστροφής.



Εικόνα 10.2: Αυτοματοποιημένο μήνυμα συναγερμού

Κεφάλαιο 11ο: Συμπεράσματα, Περιορισμοί και Μελλοντικές Επεκτάσεις

11.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία κατέδειξε επιτυχώς ότι η ενσωμάτωση τεχνολογιών του Διαδικτύου των Πραγμάτων (IoT) σε συστήματα αποθήκευσης ενέργειας δύναται να αναβαθμίσει ουσιαστικά τη διαχείριση και την προστασία τους. Η πειραματική διάταξη που υλοποιήθηκε επιβεβαίωσε ότι ένας μικροελεγκτής χαμηλού κόστους (ESP32), σε συνδυασμό με αισθητήρες ακριβείας (INA226, DS18B20), κρίνεται επαρκής για την ανάπτυξη ενός αξιόπιστου Τοπικού Συστήματος Διαχείρισης Μπαταρίας (Edge BMS). Το σύστημα απέδειξε την ικανότητά του να προστατεύει αυτόματα τη μπαταρία μολύβδου από συνθήκες βαθιάς εκφόρτισης, μέσω της έγκαιρης αποκοπής του φορτίου, συμβάλλοντας με τον τρόπο αυτό στην παράταση του κύκλου ζωής της.

Παράλληλα, σε επίπεδο λογισμικού, η αναπτυχθείσα αρχιτεκτονική επιβεβαίωσε την αποτελεσματικότητα των σύγχρονων υποδομών cloud. Η αδιάλειπτη μεταφορά δεδομένων, μέσω μορφώτου JSON και πρωτοκόλλου HTTP POST, προς το Django Backend, καθώς και η αποθήκευσή τους στη serverless βάση δεδομένων PostgreSQL (Neon), πραγματοποιήθηκαν χωρίς απώλεια πληροφορίας. Τέλος, η ανάπτυξη της διαδραστικής Διεπαφής Χρήστη (Dashboards) κατέδειξε ότι η πολύπλοκη τηλεμετρία είναι δυνατόν να οπτικοποιηθεί με τρόπο ευκόλως κατανοητό, εξυπηρετώντας τόσο τον απλό οικιακό χρήστη όσο και τον κεντρικό διαχειριστή ενός ευρύτερου στόλου εγκαταστάσεων.

Σχετικά με το EE1, τα πειραματικά αποτελέσματα του Κεφαλαίου 10 επιβεβαίωσαν ότι ο ESP32 είναι πράγματι ικανός να εκτελεί αξιόπιστη τοπική προστασία (Edge BMS): η ενεργοποίηση της αποκοπής φορτίου μέσω του κυκλώματος διπλού MOSFET πραγματοποιήθηκε αποκλειστικά βάσει τοπικής λογικής στον μικροελεγκτή, χωρίς να μεσολαβεί απόκριση από το backend, επιβεβαιώνοντας την ανεξαρτησία του μηχανισμού προστασίας από τη διαθεσιμότητα του διαδικτύου. Σχετικά με το EE2, ο συνδυασμός OCV και Coulomb Counting απέδειξε ότι μπορεί να ανιχνεύσει ποιοτικά τη φθορά του συσσωρευτή: η ταχεία πτώση τάσης υπό φορτίο (από τα 11.5V στα 10.83V εντός 5 λεπτών) αποτέλεσε σαφή ένδειξη αυξημένης εσωτερικής αντίστασης και άρα χαμηλής Κατάστασης Υγείας (SoH). Ωστόσο, όπως αναλύεται στην ενότητα 11.2, η ποσοτική ακρίβεια της μεθόδου δεν επικυρώθηκε συστηματικά έναντι οργάνου αναφοράς, με αποτέλεσμα η απάντηση στο EE2 να είναι ποιοτική και όχι πλήρως ποσοτικοποιημένη. Σχετικά με το EE3, η ενιαία αρχιτεκτονική Django απέδειξε στην πράξη ότι μπορεί να εξυπηρετήσει ταυτόχρονα τόσο τον οικιακό χρήστη (μέσω του Client Dashboard) όσο και τον διαχειριστή στόλου (μέσω του Fleet Dashboard), χωρίς διπλασιασμό κώδικα ή υποδομής, όπως παρουσιάστηκε στο Κεφάλαιο 9.

11.2 Περιορισμοί της Έρευνας

Παρά τα θετικά αποτελέσματα, η παρούσα εργασία φέρει ορισμένους περιορισμούς, οι οποίοι πρέπει να ληφθούν υπόψη κατά την αξιολόγηση των συμπερασμάτων και οριοθετούν το πεδίο ισχύος τους.

- **Περιορισμένο Μέγεθος Δείγματος:** Οι πειραματικές δοκιμές πραγματοποιήθηκαν σε έναν και μόνο συσσωρευτή VRLA και σε έναν μοναδικό κύκλο επιταχυνόμενης εκφόρτισης. Δεν πραγματοποιήθηκαν επαναλαμβανόμενες δοκιμές, με αποτέλεσμα να μην είναι δυνατή η εξαγωγή στατιστικά σημαντικών συμπερασμάτων για την ακρίβεια των εκτιμήσεων SoC/SoH.
- **Απουσία Οργάνου Αναφοράς:** Η ακρίβεια των μετρήσεων του αισθητήρα INA226 και των υπολογιζόμενων δεικτών SoC/SoH δεν συγκρίθηκε με πιστοποιημένο όργανο αναφοράς (π.χ.

εργαστηριακό όργανο υψηλής ακρίβειας ή αναλυτή μπαταριών), οπότε το πραγματικό περιθώριο σφάλματος του συστήματος παραμένει άγνωστο.

- **Εργαστηριακές, Επιταχυνόμενες Συνθήκες Δοκιμής:** Η αξιολόγηση πραγματοποιήθηκε σε ελεγχόμενο εσωτερικό περιβάλλον μέσω τεχνικών επιταχυνόμενης δοκιμής, και όχι μέσω μακροχρόνιας εγκατάστασης σε πραγματικές συνθήκες υπαίθρου. Παράγοντες όπως οι εποχιακές διακυμάνσεις θερμοκρασίας, η μακροχρόνια γήρανση σε βάθος μηνών και η μεταβλητότητα της ηλιοφάνειας δεν καλύφθηκαν.
- **Δοκιμή Δικτύου σε Ιδανικές Συνθήκες:** Αν και ένα από τα βασικά κίνητρα της εργασίας (ενότητα 1.2) είναι η κάλυψη απομακρυσμένων, εκτός δικτύου εγκαταστάσεων, όλες οι δοκιμές πραγματοποιήθηκαν μέσω σταθερού τοπικού δικτύου Wi-Fi. Η συμπεριφορά του συστήματος σε συνθήκες ασθενούς ή ασταθούς συνδεσιμότητας δεν αξιολογήθηκε πειραματικά.
- **Περιορισμένη Κλίμακα Δοκιμής Fleet Management:** Το υποσύστημα διαχείρισης στόλου παρουσιάστηκε και δοκιμάστηκε λειτουργικά, όχι όμως σε συνθήκες πραγματικής κλίμακας με πολλαπλές ταυτόχρονες συσκευές, οπότε η απόδοση της βάσης δεδομένων και του backend υπό φόρτο παραμένει αδοκίμαστη.
- **Πεδίο Εφαρμογής ως προς τη Χημεία Μπαταρίας:** Ο αλγόριθμος εκτίμησης SoC/SoH σχεδιάστηκε και επικυρώθηκε αποκλειστικά για συσσωρευτές μολύβδου-οξέος (VRLA). Η εφαρμοσιμότητά του σε άλλες χημείες (π.χ. Li-ion) δεν είναι δεδομένη χωρίς περαιτέρω προσαρμογή, όπως συζητείται και στη ενότητα 11.4.

11.3 Προτάσεις για τροποποιήσεις

Παρά την επιτυχή λειτουργία του πρωτοτύπου, κατά τη διάρκεια της ανάπτυξης και των δοκιμών εντοπίστηκαν συγκεκριμένοι περιορισμοί που επιδέχονται τροποποιήσεων σε μελλοντικές εκδόσεις:

- **Εξάρτηση από δίκτυο Wi-Fi:** Το τρέχον σύστημα απαιτεί την ύπαρξη τοπικού ασύρματου δικτύου (Router) για την αποστολή των δεδομένων. Σε πραγματικές συνθήκες, πολλά αυτόνομα φωτοβολταϊκά συστήματα βρίσκονται σε απομακρυσμένες, εκτός δικτύου (off-grid) περιοχές (π.χ. αγροτικές εγκαταστάσεις). Η αντικατάσταση ή προσθήκη μιας μονάδας επικοινωνίας GSM/LTE (π.χ. SIM800L) ή τεχνολογίας LoRaWAN θα έλυνε αυτό το πρόβλημα.
- **Ενεργειακή Κατανάλωση Μικροελεγκτή:** Αν και το ESP32 είναι αποδοτικό, η συνεχής λειτουργία του (χωρίς χρήση Deep Sleep) καταναλώνει ένα μικρό αλλά υπολογίσιμο ποσοστό ενέργειας από την ίδια τη μπαταρία που καλείται να προστατεύσει. Μια τροποποίηση του κώδικα ώστε η αποστολή δεδομένων να γίνεται περιοδικά (π.χ. κάθε 5 λεπτά) και ενδιάμεσα να μπαίνει σε κατάσταση ύπνου, θα βελτιώνει το συνολικό ενεργειακό ισοζύγιο.
- **Αντιμετώπιση Ταλάντωσης (Chattering):** Λόγω της μεγάλης πτώσης τάσης υπό φορτίο (Voltage Sag), η ενεργοποίηση του λαμπτήρα έριχνε στιγμιαία την τάση κάτω από το όριο ασφαλείας, με αποτέλεσμα ο μικροελεγκτής να αποκόπτεi το φορτίο, προκαλώντας συνεχή και γρήγορη εναλλαγή καταστάσεων (chattering). Σε επόμενη έκδοση, προτείνεται η ενσωμάτωση ενός χρονικού φίλτρου (Debounce Timer) ή η αύξηση του παραθύρου υστέρησης (Hysteresis), ώστε η τάση να πρέπει να παραμείνει χαμηλά για ένα συνεχόμενο χρονικό διάστημα πριν την αποκοπή.
- **Ελεγκτής Φόρτισης PWM:** Στην πειραματική διάταξη χρησιμοποιήθηκε ρυθμιστής τεχνολογίας PWM για λόγους κόστους. Μια αναβάθμιση σε ρυθμιστή τεχνολογίας MPPT (Maximum Power Point Tracking) θα αύξανε σημαντικά την απόδοση συλλογής ενέργειας από το φωτοβολταϊκό πάνελ.

11.4 Μελλοντικές Επεκτάσεις

Με βάση την υπάρχουσα υποδομή, το έργο παρέχει μια ισχυρή βάση για περαιτέρω έρευνα και μελλοντικές επεκτάσεις:

- **Ενσωμάτωση Μηχανικής Μάθησης (Machine Learning):** Τα ιστορικά δεδομένα που συγκεντρώνονται στη βάση δεδομένων (χιλιάδες εγγραφές τάσης, ρεύματος, θερμοκρασίας) μπορούν να αποτελέσουν το σετ εκπαίδευσης (training set) για αλγορίθμους τεχνητής

νοημοσύνης. Ένα μοντέλο πρόβλεψης (Predictive Maintenance) θα μπορούσε να προβλέψει με ακρίβεια την επερχόμενη βλάβη της μπαταρίας εβδομάδες πριν αυτή συμβεί, ειδοποιώντας τον διαχειριστή.

- **Υποστήριξη Μπαταριών Λιθίου (Li-ion / LiFePO4):** Καθώς η αγορά απομακρύνεται σταδιακά από τον μόλυβδο, ο αλγόριθμος του συστήματος μπορεί να επεκταθεί ώστε να υποστηρίζει και άλλες χημικές συστάσεις μπαταριών. Αυτό θα απαιτούσε την προσθήκη δυνατοτήτων εξισορρόπησης κελιών (Active Cell Balancing) στο κύκλωμα μέτρησης.
- **Ανάπτυξη Εγγενούς Κινητής Εφαρμογής (Native Mobile App):** Ενώ η τρέχουσα λύση βασίζεται σε μια ανταποκρινόμενη ιστοσελίδα (responsive web app), η ανάπτυξη μιας αποκλειστικής εφαρμογής για iOS και Android (π.χ. μέσω React Native ή Flutter) θα επέτρεπε την αποστολή ειδοποιήσεων Push (Push Notifications) στο κινητό του χρήστη σε πραγματικό χρόνο, σε περιπτώσεις κρίσιμων συναγερμών (Alerts).
- **Οικονομική Ανάλυση Κόστους-Οφέλους:** Μελλοντική έρευνα θα μπορούσε να περιλαμβάνει συστηματική συγκριτική ανάλυση κόστους (Bill of Materials) του προτεινόμενου συστήματος έναντι αντίστοιχων εμπορικών λύσεων BMS, καθώς και εκτίμηση του χρόνου απόσβεσης (payback period) μέσω της παράτασης του κύκλου ζωής της μπαταρίας.
- **Ενίσχυση Ασφαλείας σε Επίπεδο API:** Η μετάβαση από απλή ταυτοποίηση βάσει MAC address σε μηχανισμούς όπως τα JWT tokens, καθώς και η συστηματική δοκιμή διείσδυσης (penetration testing) του API, θα ενίσχυαν την ανθεκτικότητα του συστήματος σε κακόβουλη χρήση.
- **Μακροχρόνια Πιλοτική Εγκατάσταση:** Η ανάπτυξη του συστήματος σε πραγματική, μακροχρόνια εγκατάσταση υπαίθρου (τουλάχιστον 6-12 μηνών) θα επέτρεπε την επικύρωση του μοντέλου εκτίμησης SoH έναντι πραγματικής γήρανσης του συσσωρευτή, απαντώντας πληρέστερα στο EE2.
- **Αυτοματοποιημένη Καταμέτρηση Κύκλων (Cycle Counting / Ah Throughput):** Μελλοντικά προτείνεται η ενσωμάτωση αλγορίθμου δυναμικού υπολογισμού των ισοδύναμων πλήρων κύκλων (Equivalent Full Cycles - EFC) στο backend μέσω ολοκλήρωσης του διερχόμενου ρεύματος (Ah throughput) στον χρόνο, ώστε να προστεθεί ως επιπλέον δείκτης σωρευτικής καταπόνησης της μπαταρίας πέραν της συνολικής ενέργειας (Wh).

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] F. Colace, C. Elia, C. G. Guida, A. Lorusso, F. Marongiu, and D. Santaniello, “An IoT-based framework to protect cultural heritage buildings,” in 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 377–382, doi: 10.1109/SMARTCOMP52413.2021.00076.
- [2] M. Maksimović and M. Čosović, “Preservation of cultural heritage sites using IoT,” in 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2019, pp. 1–4, doi: 10.1109/INFOTEH.2019.8717658.
- [3] K. K. Patel and S. M. Patel, “Internet of Things-IOT: Definition, characteristics, architecture, enabling technologies, application & future challenges,” International Journal of Engineering Science and Computing, vol. 6, no. 5, pp. 6122–6127, May 2016, doi: 10.4010/2016.1482.
- [4] S. Pasricha, “Embedded systems education: Experiences with application-driven pedagogy,” IEEE Embedded Systems Letters, vol. 14, no. 4, pp. 167–170, Dec. 2022, doi: 10.1109/LES.2022.3175686.
- [5] S. Dey and T. Bera, “Design and development of a smart and multipurpose IoT embedded system device using ESP32 microcontroller,” in 2023 International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM), Roorkee, India, 2023, pp. 1–6, doi: 10.1109/ELEXCOM58812.2023.10370327.
- [6] V. V. R, A. V, K. T. P, A. K. Sunil, R. S, and H. T, “Implementation of a Smart IoT-Integrated Battery Management System with SOC, SOH and SOP Monitoring and Enhanced Fire Safety,” in 2025 4th International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), 2025, pp. 592–597. doi: 10.1109/ACCESS65134.2025.11135597.
- [7] P. Sreekanth, I. Anupalli, and P. Sudheer, “A comprehensive review of SOC, SOH, and SOP estimation techniques for PV systems,” African Journal of Biomedical Research, vol. 27, no. 6S, pp. 532–539, Dec. 2024, doi: 10.53555/AJBR.v27i6S.7065.
- [8] Codrey Electronics, “What is embedded system and how it works?,” Codrey Electronics, Sep. 1, 2022. [Online]. Available: <https://www.codrey.com/embedded-systems/embedded-systems-introduction/>. Accessed: Jul. 11, 2026.
- [9] Dhananjayan, “Embedded system architecture consulting: A design guide,” Embien, Oct. 23, 2013. [Online]. Available: <https://www.embien.com/blog/embedded-system-design-architecture>. Accessed: Jul. 11, 2026.
- [10] R. Koçi, X. Franch, P. Jovanovic, and A. Abelló, “Web API evolution patterns: A usage-driven approach,” Journal of Systems and Software, vol. 198, p. 111609, Apr. 2023, doi: 10.1016/j.jss.2023.111609.
- [11] S. Padhi, “Web API vs REST API,” Medium, Dec. 27, 2025. [Online]. Available: <https://medium.com/@shubhampadhi404/web-api-vs-rest-api-82de36b9ebb2>. Accessed: Jul. 11, 2026.
- [12] N. Sahni, J. Bose, and K. Das, “Web APIs for Internet of Things,” in 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 2175–2181, doi: 10.1109/ICACCI.2018.8554612.
- [13] Grobotronics, “Grobotronics.” [Online]. Available: <https://grobotronics.com/>. Accessed: Jul. 11, 2026.

- [14] Analog Devices, “DS18B20 programmable resolution 1-Wire digital thermometer,” Datasheet, Rev. 6, Aug. 2019. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf>. Accessed: Jul. 11, 2026.
- [15] Espressif Systems, “ESP32 Series Datasheet,” Version 4.5, Feb. 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- [16] Texas Instruments, “INA226 high-side or low-side measurement, bi-directional current and power monitor with I2C compatible interface,” SBOS547D, 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina226.pdf?ts=1788566845913&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fde-de%252FINA226. Accessed: Jul. 11, 2026.
- [17] Texas Instruments, “LM2596 SIMPLE SWITCHER® power converter 150-kHz 3-A step-down voltage regulator,” SNVS120I, 2020. [Online]. Available: <https://www.ti.com/lit/ds/symlink/lm2596.pdf>. Accessed: Jul. 11, 2026.
- [18] HellasDigital.gr, “INA226 current and power monitor module.” [Online]. Available: <https://www.hellasdigital.gr/images/detailed/15/ina226.png>. Accessed: Jul. 11, 2026.
- [19] NXP Semiconductors, “I2C-bus specification and user manual,” UM10204, Rev. 7.0, Oct. 2021. [Online]. Available: <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>. Accessed: Jul. 11, 2026.
- [20] Fritzing, “Fritzing: Electronics made easy.” [Online]. Available: <https://fritzing.org/>. Accessed: Jul. 11, 2026.
- [21] Codefinity, “Codefinity: Courses with certificates | Online learning platform,” 2026. [Online]. Available: <https://codefinity.com/courses/v2/f9161f23-e81e-4082-9642-0a0c970c68aa/0ce97d0f-ac61-4db5-9ea5-9590e9c7f14c/899e5b82-04d9-4bb5-b3a8-e18ae1cf90b8>. Accessed: Jul. 11, 2026.
- [22] Neon, “Neon: Serverless Postgres,” 2026. [Online]. Available: <https://neon.com/>. Accessed: Jul. 11, 2026.
- [23] C. Ranaweera, “Lead acid battery charging stages: Bulk, absorption & float,” Off-grid Wizard, Dec. 7, 2023. [Online]. Available: <https://offgridwizard.com/power-backup-systems/lead-acid-battery-charging-stages-bulk-absorption-float/>. Accessed: Jul. 11, 2026.
- [24] N. A. Rohman, N. F. A. Rahman, and M. A. A. M. Zainuri, “Characteristics of lead-acid and nickel metal hydride batteries in uninterruptible power supply operation,” *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 3, pp. 1520–1528, Sep. 2019, doi: 10.11591/ijpeds.v10.i3.pp1520-1528.
- [25] I. U. V. Simanjuntak, H. Heryanto, Y. Rahmawaty, and T. Manurung, “Performance analysis of VRLA battery for DC load at telecommunication base station,” *ELKHA: Jurnal Teknik Elektro*, vol. 13, no. 2, pp. 148–154, Oct. 2021, doi: 10.26418/elkha.v13i2.49202.
- [26] J. Loukil, F. Masmoudi, and N. Derbel, “State of charge estimation of lead acid battery using a Kalman filter,” in 2017 14th International Multi-Conference on Systems, Signals & Devices (SSD), Sfax, Tunisia, 2017, pp. 308–312, doi: 10.1109/SSD.2017.8167026.
- [27] R. Yu, G. Liu, L. Xu, Y. Ma, H. Wang, and C. Hu, “Review of degradation mechanism and health estimation method of VRLA battery used for standby power supply in power system,” *Coatings*, vol. 13, no. 3, p. 485, Feb. 2023, doi: 10.3390/coatings13030485.

- [28] AliExpress, "AliExpress." [Online]. Available: <https://www.aliexpress.com/>. Accessed: Jul. 11, 2026.
- [29] S. V. Salunke and A. Ouda, "A Performance Benchmark for the PostgreSQL and MySQL Databases," *Future Internet*, vol. 16, no. 10, p. 382, Oct. 2024.
- [30] PostgreSQL Global Development Group, "PostgreSQL: The World's Most Advanced Open-Source Relational Database," 2026. [Online]. Διαθέσιμο: <https://www.postgresql.org/>
- [31] V. Kjorveziroski, S. Filiposka, and V. Trajkovik, "IoT Serverless Computing at the Edge: A Systematic Mapping Review," *Computers*, vol. 10, no. 10, p. 130, Oct. 2021.
- [32] F. Li, "Cloud-native database systems at Alibaba: Opportunities and Challenges," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 2263–2272, Aug. 2019.
- [33] A. Verbitski, A. Gupta, D. Saha, M. Brahmadesam, K. Gupta, R. Mittal, S. Krishnamurthy, S. Maurice, T. Kharatishvili, and X. Bao, "Amazon Aurora: Design considerations for high throughput cloud-native relational databases," in *Proc. 2017 ACM Int. Conf. Manage. Data (SIGMOD)*, 2017, pp. 1041–1052, doi: 10.1145/3035918.3056101.
- [34] Waveshare, "NodeMCU-32S, ESP32 Development Board, WiFi / Bluetooth / UART / Keys / Extension Headers," Waveshare.com. [Online]. Available: <https://www.waveshare.com/nodemcu-32s.htm>.
- [35] P. Sethi and S. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–25, 2017, doi: 10.1155/2017/9324035.

ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ

- MicroPython

Αρχείο main.py

Είναι αυτό που τρέχει στον ESP32

main.py

```
import network
import urequests as requests
import time
import gc
from machine import Pin, I2C
import ustruct
import onewire
import ds18x20
import secrets

# --- Ρυθμίσεις συστήματος ---
URL = 'http://192.168.1.15:8000/api/sensor-data/'
MAC_ADDRESS = "AA:BB:CC:44:55:66"

# --- Ρυθμίσεις hardware ---
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
INA226_ADDR = 0x40
R_TOTAL = 0.0014778 # Αντίσταση shunt για τον υπολογισμό ρεύματος

ds_pin = Pin(4)
ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin)) # Αισθητήρας θερμοκρασίας
mosfet = Pin(15, Pin.OUT) # Έλεγχος φορτίου (relay/mosfet)

def read_ina_register(reg):
    # Ανάγνωση 2 bytes από τον αισθητήρα ρεύματος/τάσης INA226 (signed, big-endian)
    data = i2c.readfrom_mem(INA226_ADDR, reg, 2)
    return ustruct.unpack('>h', data)[0]
```

```

def connect_wifi():
    # Σύνδεση στο Wi-Fi με timeout· επανεκκίνηση αν αποτύχει
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print("Σύνδεση στο Wi-Fi...", end="")
        wlan.connect(secrets.WIFI_SSID, secrets.WIFI_PASS)
        timeout = 20
        while not wlan.isconnected() and timeout > 0:
            time.sleep(1)
            print(".", end="")
            timeout -= 1
        if not wlan.isconnected():
            print("\n[ERROR] Αποτυχία σύνδεσης στο Wi-Fi. Επανεκκίνηση...")
            import machine
            machine.reset()
        print(f"\n[NETWORK] Επιτυχής σύνδεση! IP: {wlan.ifconfig()[0]}")
    return wlan

def get_averaged_ina_readings(samples=5):
    # Oversampling: παίρνουμε πολλαπλές μετρήσεις και επιστρέφουμε τον μέσο όρο για μείωση
    # θορύβου
    sum_v, sum_i = 0.0, 0.0
    for _ in range(samples):
        raw_bus = read_ina_register(0x02)
        sum_v += raw_bus * 0.00125 # Μετατροπή raw τιμής σε Volts

        raw_shunt = read_ina_register(0x01)
        sum_i += (raw_shunt * 0.0000025) / R_TOTAL # Μετατροπή raw τιμής σε Amps
    time.sleep_ms(50)

```

```

return round(sum_v / samples, 2), round(sum_i / samples, 3)

# --- Κύριο πρόγραμμα ---
def main():
    wlan = connect_wifi()
    print("[SYSTEM] Σάρωση DS18B20...")
    roms = ds_sensor.scan()
    if not roms:
        print("[WARNING] Δεν βρέθηκε DS18B20. Η θερμοκρασία θα καταγράφεται ως 0.0°C")

    print("\n--- Σύστημα IoT Ενεργό: Παρακολούθηση Φ/Β Μπαταρίας ---")
    mosfet.value(0) # Το φορτίο ξεκινά κλειστό

    while True:
        try:
            # Επανασύνδεση αν χαθεί το Wi-Fi
            if not wlan.isconnected():
                print("[NETWORK] Απώλεια σύνδεσης. Επανασύνδεση...")
                connect_wifi()

            # Ανάγνωση θερμοκρασίας (αν υπάρχει αισθητήρας)
            if roms:
                ds_sensor.convert_temp()
                time.sleep_ms(750) # Χρόνος αναμονής μετατροπής DS18B20
                temp_c = round(ds_sensor.read_temp(roms[0]), 1)
            else:
                temp_c = 0.0
                time.sleep_ms(750)

            # Oversampling για ακρίβεια
            real_voltage, real_current = get_averaged_ina_readings()

```

```

if real_voltage < 11.5 or temp_c > 45.0:
    mosfet.value(0)
elif real_voltage > 12.5 and temp_c < 40.0:
    mosfet.value(1)

lamp_status = "ON" if mosfet.value() == 1 else "OFF"

# Πακέτο δεδομένων προς αποστολή στο backend
payload = {
    "mac_address": MAC_ADDRESS,
    "voltage": real_voltage,
    "current": real_current,
    "temperature": temp_c
}

print(f"[DATA] V: {real_voltage}V | I: {real_current}A | T: {temp_c}°C | Load: {lamp_status} ->", end=" ")

response = None
try:
    response = requests.post(URL, json=payload)
    print(f"Status: {response.status_code}")
except Exception as req_e:
    print(f"[HTTP ERROR] Αποτυχία αποστολής: {req_e}")
finally:
    if response:
        response.close() # Απαραίτητο για αποφυγή διαρροής μνήμης

gc.collect() # Χειροκίνητο garbage collection (σημαντικό σε microcontrollers)

except Exception as e:
    print(f"\n[ERROR] Σφάλμα βρόχου: {e}")
    time.sleep(2) # Μικρή παύση πριν την επόμενη προσπάθεια

```

```
time.sleep(5) # Ρυθμός ανανέωσης δεδομένων (κάθε 5 δευτερόλεπτα)

if __name__ == "__main__":
    main()
```

- Python

Αρχεία models.py, views.py

models.py

```
from django.db import models
from django.contrib.auth.models import User

class Device(models.Model):
    # MAC address ως Primary Key, εξασφαλίζει μοναδικότητα ανά φυσική συσκευή
    mac_address = models.CharField(max_length=17, primary_key=True, help_text="π.χ. AA:BB:CC:DD:EE:FF")
    name = models.CharField(max_length=100, help_text="Ένα όνομα, π.χ. 'Μπαταρία Ταράτσας")

    # ManyToMany: ένας χρήστης μπορεί να έχει πρόσβαση σε πολλές συσκευές και αντίστροφα
    users = models.ManyToManyField(User, related_name='allowed_devices', blank=True)

    def __str__(self):
        return f'{self.name} ({self.mac_address})'

# Πίνακας Μετρήσεων (Measurements)
class Measurement(models.Model):
    # Χρόνος λήψης της μέτρησης (όχι Primary Key, αυτόματο BigAutoField id χρησιμοποιείται ως PK)
    timestamp = models.DateTimeField()

    # Σύνδεση της μέτρησης με τη συσκευή (ESP32) που την απέστειλε
```

```

device = models.ForeignKey(Device, on_delete=models.CASCADE,
related_name='measurements')

#Τιμές αισθητήρων
voltage = models.FloatField(verbose_name="Τάση (V)")
current = models.FloatField(verbose_name="Ρεύμα (A)")
temperature = models.FloatField(verbose_name="Θερμοκρασία (°C)")

# Παράγωγα/υπολογισμένα μεγέθη κατάστασης μπαταρίας
SoC = models.FloatField(verbose_name="State of Charge (%)", null=True, blank=True)
SoH = models.FloatField(verbose_name="State of Health (%)", null=True, blank=True)
power = models.FloatField(verbose_name="Ισχύς (W)", null=True, blank=True)

class Meta:
    ordering = ['-timestamp'] # προεπιλεγμένη ταξινόμηση: πιο πρόσφατη μέτρηση πρώτη

def __str__(self):
    return f"{self.device.mac_address} - {self.timestamp}"

```

views.py

```

import os
import json
import threading
from datetime import timedelta

from django.utils import timezone
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.shortcuts import render, redirect
from django.contrib.auth import login
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from django import forms

```

```

from django.contrib.auth.decorators import login_required
from django.core.cache import cache

from .models import Device, Measurement

#=====
# 1. ΑΣΥΓΧΡΟΝΗ ΑΠΟΣΤΟΛΗ ΕΙΔΟΠΟΙΗΣΕΩΝ (BACKGROUND EMAIL THREAD)
#=====

def send_email_in_background(subject, message, from_email, recipient_list):
    """
    Βοηθητική συνάρτηση που εκτελείται ασύγχρονα σε ανεξάρτητο νήμα (Worker Thread).
    Αποτρέπει την καθυστέρηση (latency) στην επιστροφή της απάντησης HTTP προς
    τον μικροελεγκτή ESP32 κατά την αποστολή email μέσω του διακομιστή SMTP (Gmail).
    """
    try:
        from django.core.mail import send_mail
        send_mail(
            subject,
            message,
            from_email,
            recipient_list,
            fail_silently=False,
        )
        print(f'[MAIL SUCCESS] Background email sent successfully: {subject}')
    except Exception as e:
        print(f'[MAIL ERROR] Background Error sending email: {e}')

#=====
# 2. ΠΙΝΑΚΑΣ ΕΛΕΓΧΟΥ ΠΙΛΑΤΗ (CLIENT DASHBOARD)
#=====

@login_required

```

```

def home(request):
    """
    Προβολή του εξατομικευμένου Πίνακα Ελέγχου (Client Dashboard).
    Ανακτά τα ιστορικά δεδομένα της επιλεγμένης συσκευής, υπολογίζει δυναμικά
    την Εσωτερική Αντίσταση (IR), τη Συνολική Παραγόμενη Ενέργεια (Wh), και
    τροφοδοτεί τη βιβλιοθήκη Chart.js μέσω JSON σειριοποίησης.
    """
    selected_device_mac = request.GET.get('device')

    # Εάν ο συνδεδεμένος χρήστης είναι διαχειριστής και δεν έχει επιλέξει συγκεκριμένο κόμβο,
    # ανακατευθύνεται αυτόματα στο Κέντρο Διαχείρισης Στόλου (Fleet Dashboard)
    if request.user.is_superuser and not selected_device_mac:
        return redirect('fleet_dashboard')

    # Διαχωρισμός δικαιωμάτων: Ο superuser βλέπει όλους τους κόμβους, ο απλός χρήστης μόνο τους
    εξουσιοδοτημένους
    if request.user.is_superuser:
        user_devices = Device.objects.all()
    else:
        user_devices = request.user.allowed_devices.all()

    # Επιλογή της ζητούμενης συσκευής ή προεπιλογή της πρώτης διαθέσιμης
    if selected_device_mac:
        device = user_devices.filter(pk=selected_device_mac).first()
    else:
        device = user_devices.first()

    # Αρχικοποίηση δομών δεδομένων για τις χρονοσειρές των διαγραμμάτων (Chart.js)
    timestamps, soc_data, soh_data = [], [], []
    voltage_data, current_data, temp_data = [], [], []
    power_data, dod_data, ir_data = [], [], []
    total_energy_wh = 0.0

```

```

if device:

    # Ανάκτηση των τελευταίων 200 μετρήσεων για απεικόνιση των πρόσφατων τάσεων
    measurements = Measurement.objects.filter(device=device).order_by('-timestamp')[:200]

    last_v = None
    last_i = None

    last_valid_ir = 40.0 # Τυπική ονομαστική εσωτερική αντίσταση υγιούς μπαταρίας VRLA 4Ah (σε
mΩ)

    # Αντιστροφή για χρονολογική σειρά (από το παλαιότερο προς το νεότερο)
    for m in reversed(measurements):

        timestamps.append(m.timestamp.strftime('%H:%M:%S'))

        # Δείκτες Φόρτισης και Υγείας
        current_soc = m.soc if m.soc is not None else 0.0
        soc_data.append(current_soc)
        dod_data.append(round(100.0 - current_soc, 1)) # Βάθος Εκφόρτισης: DoD = 100 - SoC
        soh_data.append(m.soh if m.soh is not None else 100.0)

        # Φυσικά τηλεμετρικά μεγέθη
        voltage_data.append(m.voltage)
        current_data.append(m.current)
        temp_data.append(m.temperature)
        power_data.append(m.power if m.power is not None else 0.0)

        # --- Δυναμικός Υπολογισμός Εσωτερικής Αντίστασης (Internal Resistance - IR) ---
        # Εφαρμογή του νόμου του Ohm βάσει της στιγμιαίας πτώσης τάσης υπό φορτίο (Voltage Sag)
        if last_v is not None and last_i is not None:
            delta_v = abs(last_v - m.voltage)
            delta_i = abs(last_i - m.current)

            # Φιλτράρισμα θορύβου: υπολογισμός μόνο σε ουσιαστικές μεταβολές ρεύματος (> 0.5A)
            if delta_i > 0.5:
                current_ir = (delta_v / delta_i) * 1000.0 # Μετατροπή σε mΩ

```

```

# Απόρριψη μη φυσικών ακραίων τιμών (εύρος φίλτρου: 10 mΩ έως 1500 mΩ)
if 10.0 <= current_ir <= 1500.0:
    last_valid_ir = round(current_ir, 1)

ir_data.append(last_valid_ir)

last_v = m.voltage
last_i = m.current

# --- Υπολογισμός Συνολικής Παραγόμενης Ενέργειας (Total Energy Wh) ---
# Εκτελείται ΕΞΩ από τον βρόχο των 200 μετρήσεων για βέλτιστη απόδοση (αριθμητική
ολοκλήρωση Simpson/Trapezoidal)
all_measurements = Measurement.objects.filter(device=device).order_by('timestamp')
prev_time = None
for m in all_measurements:
    if prev_time is not None:
        dt_hours = (m.timestamp - prev_time).total_seconds() / 3600.0

        # Αγνόηση αδικαιολόγητα μεγάλων κενών επικοινωνίας/αποσύνδεσης (> 6 λεπτά)
        if dt_hours < 0.1:
            p = m.power if m.power is not None else abs(m.voltage * m.current)
            total_energy_wh += p * dt_hours
            prev_time = m.timestamp

total_energy_wh = round(total_energy_wh, 2)

# Συσκευασία δεδομένων στο λεξικό context με ασφαλή σειριοποίηση JSON για το frontend
context = {
    'devices': user_devices,
    'selected_device': device,
    'total_energy_wh': total_energy_wh,
    'timestamps': json.dumps(timestamps),
    'soc_data': json.dumps(soc_data),
    'soh_data': json.dumps(soh_data),
    'voltage_data': json.dumps(voltage_data),

```

```

        'current_data': json.dumps(current_data),
        'power_data': json.dumps(power_data),
        'temp_data': json.dumps(temp_data),
        'dod_data': json.dumps(dod_data),
        'ir_data': json.dumps(ir_data),
    }

```

```

    return render(request, 'home.html', context)

```

```

#=====

```

```

# 3. ΚΕΝΤΡΟ ΔΙΑΧΕΙΡΙΣΗΣ ΣΤΟΛΟΥ (FLEET DASHBOARD)

```

```

#=====

```

```

@login_required

```

```

def fleet_dashboard(request):

```

```

    """

```

Συγκεντρωτική οθόνη εποπτείας στόλου (Fleet Management).

Προσβάσιμη αποκλειστικά από διαχειριστές (superusers) για τον έλεγχο της κατάστασης συνδεσιμότητας (Online/Offline) και την ανίχνευση ενεργών σφαλμάτων σε πραγματικό χρόνο.

```

    """

```

```

    if not request.user.is_superuser:

```

```

        return redirect('/')

```

```

    all_devices = Device.objects.all()

```

```

    total_devices = all_devices.count()

```

```

    online_count = 0

```

```

    offline_count = 0

```

```

    alerts = []

```

```

    device_status_list = []

```

```

    # Χρονικό κατώφλι ενεργής συσκευής (1 ώρα χωρίς τηλεμετρία θεωρείται Offline)

```

```

    time_threshold = timezone.now() - timedelta(hours=1)

```

```

for device in all_devices:

    latest = Measurement.objects.filter(device=device).order_by('-timestamp').first()
    is_online = False

    if latest:
        if latest.timestamp >= time_threshold:
            is_online = True
            online_count += 1
        else:
            offline_count += 1

    # Έλεγχος ορίων ασφαλείας για οπτικές ειδοποιήσεις στον πίνακα διαχειριστή
    if latest.temperature > 45.0:
        alerts.append({'device': device, 'type': 'danger', 'msg': f'Υπερθέρμανση: {latest.temperature}°C'})
    if latest.voltage < 11.5:
        alerts.append({'device': device, 'type': 'warning', 'msg': f'Κρίσιμη Τάση: {latest.voltage}V (Βαθιά Εκφόρτιση)'})
    if latest.soc is not None and latest.soc < 20.0:
        alerts.append({'device': device, 'type': 'warning', 'msg': f'Χαμηλή Μπαταρία: {latest.soc}%'})
    else:
        offline_count += 1 # Συσκευή χωρίς καμία καταγεγραμμένη μέτρηση

    device_status_list.append({
        'device': device,
        'is_online': is_online,
        'latest': latest
    })

context = {
    'total_devices': total_devices,
    'online_count': online_count,

```

```

        'offline_count': offline_count,
        'alerts': alerts,
        'device_status_list': device_status_list,
    }
    return render(request, 'fleet_dashboard.html', context)

```

```

=====

```

4. ΔΙΑΧΕΙΡΙΣΗ ΧΡΗΣΤΩΝ (REGISTRATION)

```

=====

```

```

class CustomUserCreationForm(UserCreationForm):
    """Επέκταση της πρότυπης φόρμας εγγραφής ώστε το Email να είναι υποχρεωτικό πεδίο."""
    email = forms.EmailField(required=True, label="Email")

    class Meta(UserCreationForm.Meta):
        model = User
        fields = ('username', 'email')

def register(request):
    """Διαχείριση εγγραφής νέων χρηστών με αυτόματη σύνδεση μετά την επιτυχή υποβολή."""
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('/')
        else:
            form = CustomUserCreationForm()

    return render(request, 'registration/register.html', {'form': form})

```

```

#=====
# 5. REST API ENDPOINT: ΛΗΨΗ ΤΗΛΕΜΕΤΡΙΑΣ & ΑΛΓΟΡΙΘΜΟΙ SoC / SoH
#=====

BATTERY_CAPACITY_AH = 4.0 # Ονομαστική χωρητικότητα της μπαταρίας VRLA (Ah)

@csrf_exempt # Απαλλαγή CSRF καθώς η κλήση προέρχεται απευθείας από τον μικροελεγκτή ESP32
def receive_sensor_data(request):
    """
    Κεντρικό REST API Endpoint (HTTP POST).

    Παραλαμβάνει το πακέτο JSON από τον ESP32, επικυρώνει την ταυτότητα της συσκευής (Device
    Auth),
    εκτελεί τον υβριδικό αλγόριθμο υπολογισμού SoC (OCV + Coulomb Counting) και το μοντέλο
    υποβάθμισης SoH, καταχωρεί τη μέτρηση στη βάση PostgreSQL και διαχειρίζεται
    το Σύστημα Συναγερμών (Email Alerts με μηχανισμό Cooldown).
    """
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            mac = data.get('mac_address')

            # Αυθεντικοποίηση συσκευής βάσει διεύθυνσης MAC
            try:
                device = Device.objects.get(mac_address=mac)
            except Device.DoesNotExist:
                return JsonResponse({'error': 'Unknown device. MAC address not found.'}, status=404)

            current_voltage = float(data.get('voltage'))
            current_amps = float(data.get('current'))
            current_temp = float(data.get('temperature'))

            # Φιλτράρισμα θορύβου μηδενικού ρεύματος και τάσης
            if abs(current_amps) < 0.015:
                current_amps = 0.0

```

```

if current_voltage < 1.0:
    current_voltage = 0.0

power_watts = current_voltage * current_amps # P = V * I
now = timezone.now()

# Ανάκτηση της αμέσως προηγούμενης μέτρησης για τον υπολογισμό διαφορικών μεγεθών
last_measurement = Measurement.objects.filter(device=device).order_by('-timestamp').first()

# --- 1. Υβριδικός Υπολογισμός State of Charge (SoC) ---
# Φάση Α: Αρχική εκτίμηση βάσει Τάσης Ανοιχτού Κυκλώματος (OCV Approximation)
if current_voltage >= 12.7:
    new_soc = 100.0
elif current_voltage <= 11.5:
    new_soc = 0.0
else:
    new_soc = ((current_voltage - 11.5) / 1.2) * 100.0

new_soh = 100.0

# Φάση Β: Δυναμική παρακολούθηση μέσω Coulomb Counting εφόσον υπάρχει ιστορικό
if last_measurement and last_measurement.soc is not None:
    time_diff_seconds = (now - last_measurement.timestamp).total_seconds()
    time_diff_hours = time_diff_seconds / 3600.0
    battery_capacity = getattr(device, 'capacity_ah', BATTERY_CAPACITY_AH)

    # Ολοκλήρωση ρεύματος στον χρόνο: ΔSoC = (I * Δt / C) * 100
    delta_soc = ((current_amps * time_diff_hours) / battery_capacity) * 100.0
    new_soc = last_measurement.soc + delta_soc
    new_soc = max(0.0, min(100.0, new_soc)) # Περιορισμός εντός φυσικού εύρους [0%, 100%]

    new_soh = last_measurement.soh if last_measurement.soh else 100.0
else:

```

```

new_soh = 100.0 # Αρχικοποίηση για νέα εγκατάσταση

# --- 2. Εμπειρικό Μοντέλο Υποβάθμισης State of Health (SoH) ---
# Ποινή βαθιάς εκφόρτισης (σουλφάτωση κάτω από 11.5V)
if current_voltage < 11.5:
    new_soh -= 0.005

# Ποινή θερμικής καταπόνησης (επιτάχυνση διάβρωσης άνω των 35°C κατά Arrhenius)
if current_temp > 35.0:
    new_soh -= 0.001 * (current_temp - 35.0)

# Ποινή υψηλού ρεύματος λειτουργίας (> 0.25C για μπαταρία 4Ah)
if abs(current_amps) > 1.0:
    new_soh -= 0.001

new_soh = max(0.0, min(100.0, new_soh))

# Καταχώριση νέας εγγραφής τηλεμετρίας στη βάση PostgreSQL (Neon)
Measurement.objects.create(
    timestamp=now,
    device=device,
    voltage=current_voltage,
    current=current_amps,
    power=round(power_watts, 2),
    temperature=current_temp,
    soc=round(new_soc, 3),
    soh=round(new_soh, 2)
)

# --- 3. Σύστημα Συναγερμών (Alert Engine & Cooldown) ---
alert_message = None
alert_subject = None

```

```

# Ιεράρχηση κρισιμότητας σφαλμάτων
if current_temp > 45.0:
    alert_subject = f"DANGER: Battery Overheating ({device.name})"
    alert_message = f"Temperature has reached {current_temp}°C! Please check the system
immediately."
elif current_voltage < 11.5:
    alert_subject = f"DANGER: Deep Discharge ({device.name})"
    alert_message = f"Voltage has dropped to {current_voltage}V. Risk of irreversible damage."
elif new_soc < 20.0:
    alert_subject = f"WARNING: Low Battery ({device.name})"
    alert_message = f"Battery level has dropped to {new_soc}%. The light should remain off."

# Μηχανισμός αποτροπής επαναλαμβανόμενων ειδοποιήσεων (Spam Prevention μέσω Cache
Cooldown)
if alert_message:
    cache_key = f"alert_cooldown_{device.mac_address}"

# Cooldown διάρκειας 2 ωρών (7200 δευτερόλεπτα) ανά συσκευή
if not cache.get(cache_key):
    cache.set(cache_key, True, 7200)

# Ανάκτηση διαπιστευτηρίων αποστολέα/παραλήπτη από μεταβλητές περιβάλλοντος
(.env)
from_email = os.getenv('EMAIL_HOST_USER_1')
recipient_list = [os.getenv('EMAIL_USER_2')]

# Εκκίνηση ασύγχρονου νήματος αποστολής
email_thread = threading.Thread(
    target=send_email_in_background,
    args=(alert_subject, alert_message, from_email, recipient_list)
)
email_thread.start()
print(f"[ALERT TRIGGERED] Started background thread for email: {alert_subject}")

```

```

return JsonResponse({'status': 'Success! Data has been saved.'}, status=201)

except Exception as e:

    return JsonResponse({'error': str(e)}, status=400)

return JsonResponse({'error': 'Only POST requests are allowed.'}, status=405)

```

- Κώδικας Frontend (UI & Chart.js)

Αρχείο home.html

home.html

```

{% extends "base.html" %}

{% load static %}

{% block content %}

<div class="container">

    <h1 class="dashboard-title">

        Dashboard

    </h1>

    <p class="dashboard-welcome">

        Hello <b>{{ user.username }}</b>!

    </p>

    <!--

===== -->

    <!-- 1. ΕΝΟΤΗΤΑ ΕΠΙΛΟΓΗΣ ΣΥΣΚΕΥΗΣ (Device Selection)          -->

    <!-- Δυναμική προβολή των καταχωρημένων κόμβων-μπαταριών του χρήστη.  -->

    <!-- Η παράμετρος 'device' μεταβιβάζεται μέσω HTTP GET request στο view. -->

    <!--

===== -->

    <div class="card dashboard-card">

        <h2 class="device-title">Available Devices</h2>

```

```

<div class="device-selection-container">
  {% for d in devices %}
    <a href="?device={{ d.pk }}" class="btn-action btn-device {% if selected_device and
selected_device.pk == d.pk %}active{% endif %}">
      {{ d.name }}
      <span class="device-mac">{{ d.mac_address }}</span>
    </a>
  {% endfor %}

  {% if not devices %}
    <p class="text-muted">No devices available.</p>
  {% endif %}
</div>
</div>

<!--
===== -->

<!-- 2. ΠΡΑΓΜΑΤΙΚΕΣ ΧΡΟΝΟΣΕΙΡΕΣ ΤΗΛΕΜΕΤΡΙΑΣ (Live Charts - Chart.js) -->
<!-- Συνοπτική απεικόνιση των βασικών ηλεκτρικών και θερμικών παραμέτρων. -->
<!-- Κάθε κάρτα είναι clickable και ενεργοποιεί το αναδυόμενο παράθυρο. -->
<!--
===== -->

<h2 class="chart-section-title">Live Charts</h2>
<div class="charts-row-3">
  <!-- Κάρτα Γραφήματος: Τάση Συσσωρευτή (V) -->
  <div class="card dashboard-card clickable-card" onclick="openModal('voltage')" title="Κλικ για
μεγέθυνση">
    <canvas id="voltageChart"></canvas>
  </div>

  <!-- Κάρτα Γραφήματος: Ρεύμα Φόρτισης/Εκφόρτισης (A) -->
  <div class="card dashboard-card clickable-card" onclick="openModal('current')" title="Κλικ για
μεγέθυνση">

```

```

        <canvas id="currentChart"></canvas>

    </div>

    <!-- Κάρτα Γραφήματος: Θερμοκρασία Μπαταρίας (°C) -->
    <div class="card dashboard-card clickable-card" onclick="openModal('temp')" title="Κλικ για
μεγέθυνση">
        <canvas id="tempChart"></canvas>
    </div>
</div>

<!--
===== -->

<!-- 3. ΒΑΣΙΚΟΙ ΔΕΙΚΤΕΣ ΑΠΟΔΟΣΗΣ (Key Performance Indicators - KPIs) -->
<!-- Άμεση εποπτεία (glanceability) της Κατάστασης Φόρτισης (SoC), -->
<!-- της Κατάστασης Υγείας (SoH) και της Στιγμιαίας Ηλεκτρικής Ισχύος (W). -->
<!--
===== -->

<div class="charts-row-3">
    <!-- Δείκτης SoC (%) με δυναμική μπάρα προόδου -->
    <div class="card dashboard-card clickable-card kpi-card" onclick="openModal('soc')"
title="Κλικ για ιστορικό γράφημα">
        <div class="chart-label">State of Charge (SoC)</div>
        <div id="socValue" class="display-number text-success">--%</div>
        <div class="progress-bar-bg">
            <div id="socBar" class="progress-bar-fill" style="background-color: #2ECC71; width:
0%;"></div>
        </div>
    </div>

    <!-- Δείκτης SoH (%) με δυναμική μπάρα υποβάθμισης -->
    <div class="card dashboard-card clickable-card kpi-card" onclick="openModal('soh')"
title="Κλικ για ιστορικό γράφημα">
        <div class="chart-label">State of Health (SoH)</div>
        <div id="sohValue" class="display-number text-purple">--%</div>

```

```

<div class="progress-bar-bg">
  <div id="sohBar" class="progress-bar-fill" style="background-color: #9B59B6; width:
0%;"></div>
</div>
</div>

<!-- Στιγμιαία Ισχύς:  $P = V \times I$  (W) -->
<div class="card dashboard-card clickable-card kpi-card" onclick="openModal('power')"
title="Κλικ για ιστορικό γράφημα">
  <div class="chart-label">Power (W)</div>
  <div id="powerValue" class="display-number text-warning">-- W</div>
</div>
</div>

<!--
===== -->

<!-- 4. ΕΞΕΙΔΙΚΕΥΜΕΝΟΙ ΔΕΙΚΤΕΣ ΥΓΕΙΑΣ & ΚΑΤΑΠΙΟΝΗΣΗΣ (Battery Health) -->
<!-- Υπολογιζόμενες μετρικές γήρανσης: Σωρευτική Ενέργεια, DoD και IR. -->
<!--
===== -->

<h2 class="chart-section-title" style="margin-top: 30px; color: #ffffff;">Battery Health
Metrics</h2>
<div class="charts-row-3">
  <!-- Συνολική Παραγόμενη/Καταναλωθείσα Ενέργεια (Wh) μέσω αριθμητικής ολοκλήρωσης --
>
  <div class="card dashboard-card kpi-card">
    <div class="chart-label">Total Energy (Wh)</div>
    <div class="display-number text-primary">{{ total_energy_wh }}</div>
    <div style="font-size: 0.9em; color: #888; margin-top: 5px;">Accumulated Output</div>
  </div>

  <!-- Τρέχον Βάθος Εκφόρτισης:  $DoD = 100 - SoC$  (%) -->
  <div class="card dashboard-card kpi-card">
    <div class="chart-label">Depth of Discharge (DoD)</div>

```

```

<div id="dodValue" class="display-number text-warning">--%</div>

<div style="font-size: 0.9em; color: #888; margin-top: 5px;">Current DoD</div>

</div>

<!-- Δυναμική Εσωτερική Αντίσταση:  $IR = \Delta V / \Delta I$  (mΩ) -->

<div class="card dashboard-card clickable-card kpi-card" onclick="openModal('ir')" title="Κλικ
για ιστορικό γράφημα">

  <div class="chart-label">Internal Resistance (IR)</div>

  <div id="irValue" class="display-number text-danger">-- mΩ</div>

  <div style="font-size: 0.9em; color: #888; margin-top: 5px;">Status: Calculated</div>

</div>

</div>

<!-- =====
-->

<!-- 5. ΑΝΑΔΥΟΜΕΝΟ ΠΑΡΑΘΥΡΟ (Modal Dialog) ΓΙΑ ΛΕΠΤΟΜΕΡΗ ΓΡΑΦΗΜΑΤΑ -->

<!-- Εμφανίζει μεγεθυμένη προβολή του επιλεγμένου μεγέθους με κλείδωμα -->
<!-- των αξόνων κλίμακας ανάλογα με τον τύπο της μετρικής. -->

<!-- =====
-->

<div id="chartModal" class="modal">

  <div class="modal-content">

    <span class="close-btn" onclick="closeModal()">&times;</span>

    <div class="modal-canvas-wrapper">

      <canvas id="modalCanvas"></canvas>

    </div>

  </div>

</div>

</div>

<!-- Φόρτωση της βιβλιοθήκης Chart.js μέσω επίσημου CDN -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

```

```

<script>

// =====

// A. ΑΝΑΚΤΗΣΗ ΚΑΙ ΑΠΟΣΕΙΡΙΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ (JSON Data Parsing)
// Ανάκτηση των σειριοποιημένων πινάκων που δημιουργήθηκαν στο Django view.
// Το φίλτρο '|safe' αποτρέπει το HTML escaping των χαρακτήρων JSON.
// =====

const labels = JSON.parse('{{ timestamps|safe }}');
const voltageData = JSON.parse('{{ voltage_data|safe }}');
const currentData = JSON.parse('{{ current_data|safe }}');
const tempData = JSON.parse('{{ temp_data|safe }}');
const socData = JSON.parse('{{ soc_data|safe }}');
const sohData = JSON.parse('{{ soh_data|safe }}');
const powerData = JSON.parse('{{ power_data|safe }}');
const dodData = JSON.parse('{{ dod_data|safe }}');
const irData = JSON.parse('{{ ir_data|safe }}');

// Καθολικές προεπιλογές γραφικών στοιχείων Chart.js (Dark Mode Θέμα)
Chart.defaults.color = '#E0E0E0';
Chart.defaults.borderColor = 'rgba(255, 255, 255, 0.1)';

// =====

// B. ΑΡΧΙΚΟΠΟΙΗΣΗ ΣΥΝΟΠΤΙΚΩΝ ΓΡΑΦΗΜΑΤΩΝ (Dashboard Mini Charts)
// Δημιουργία των 3 βασικών χρονοσειρών συνεχούς επιτήρησης.
// =====

new Chart(document.getElementById('voltageChart').getContext('2d'), {
  type: 'line',
  data: {
    labels: labels,
    datasets: [{
      label: 'Τάση (Voltage V)',
      data: voltageData,
      borderColor: '#3273F6',

```

```

        backgroundColor: 'rgba(50, 115, 246, 0.2)',
        borderWidth: 2,
        fill: true,
        tension: 0.4
    }]
}
});

new Chart(document.getElementById('currentChart').getContext('2d'), {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Ρεύμα (Current A)',
            data: currentData,
            borderColor: '#F39C12',
            backgroundColor: 'rgba(243, 156, 18, 0.2)',
            borderWidth: 2,
            fill: true,
            tension: 0.4
        }]
    }
});

new Chart(document.getElementById('tempChart').getContext('2d'), {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Θερμοκρασία (°C)',
            data: tempData,
            borderColor: '#E74C3C',
            backgroundColor: 'rgba(231, 76, 60, 0.2)',

```

```

        borderWidth: 2,
        fill: true,
        tension: 0.4
    }]
}
});

// =====
// Γ. ΔΥΝΑΜΙΚΗ ΕΝΗΜΕΡΩΣΗ ΤΡΕΧΟΥΣΩΝ ΕΝΔΕΙΞΕΩΝ (KPI & Progress Bars)
// Εξαγωγή του πλέον πρόσφατου δείγματος (τελευταίο στοιχείο πίνακα)
// =====

if (socData.length > 0) {
    const latestSoC = socData[socData.length - 1];
    const latestSoH = sohData[sohData.length - 1];
    const latestPower = powerData[powerData.length - 1];
    const latestDoD = dodData[dodData.length - 1];
    const latestIR = irData[irData.length - 1];

    // Μορφοποίηση κειμένου με ακρίβεια 1 δεκαδικού ψηφίου
    document.getElementById('socValue').innerText = latestSoC.toFixed(1) + '%';
    document.getElementById('sohValue').innerText = latestSoH.toFixed(1) + '%';
    document.getElementById('powerValue').innerText = latestPower.toFixed(1) + ' W';
    document.getElementById('dodValue').innerText = latestDoD.toFixed(1) + '%';
    document.getElementById('irValue').innerText = latestIR.toFixed(1) + ' mΩ';

    // Ρύθμιση πλάτους ποσοστιαίων μπαρών προόδου
    document.getElementById('socBar').style.width = latestSoC + '%';
    document.getElementById('sohBar').style.width = latestSoH + '%';
}

// =====
// Δ. ΔΙΑΧΕΙΡΙΣΗ ΑΝΑΔΥΟΜΕΝΟΥ ΠΑΡΑΘΥΡΟΥ (Modal Chart Controller)
// Δυναμική κατασκευή/καταστροφή αντικειμένου Chart για εξοικονόμηση μνήμης

```

```

// =====
let modalChartInstance = null;

function openModal(chartType) {
    document.getElementById('chartModal').style.display = 'flex';

    // Καταστροφή προηγούμενου instance για αποτροπή flicking και memory leak
    if (modalChartInstance) {
        modalChartInstance.destroy();
    }

    let mLabel, mData, mBorder, mBg, isPercentage = false;

    // Παραμετροποίηση δεδομένων, χρωμάτων και κλιμάκωσης ανά επιλογή
    if (chartType === 'voltage') {
        mLabel = 'Τάση (V)'; mData = voltageData; mBorder = '#3273F6'; mBg = 'rgba(50, 115, 246, 0.2)';
    } else if (chartType === 'current') {
        mLabel = 'Ρεύμα (A)'; mData = currentData; mBorder = '#F39C12'; mBg = 'rgba(243, 156, 18, 0.2)';
    } else if (chartType === 'temp') {
        mLabel = 'Θερμοκρασία (°C)'; mData = tempData; mBorder = '#E74C3C'; mBg = 'rgba(231, 76, 60, 0.2)';
    } else if (chartType === 'soc') {
        mLabel = 'SoC (%)'; mData = socData; mBorder = '#2ECC71'; mBg = 'rgba(46, 204, 113, 0.2)';
        isPercentage = true;
    } else if (chartType === 'soh') {
        mLabel = 'SoH (%)'; mData = sohData; mBorder = '#9B59B6'; mBg = 'rgba(155, 89, 182, 0.2)';
        isPercentage = true;
    } else if (chartType === 'power') {
        mLabel = 'Ισχύς (W)'; mData = powerData; mBorder = '#F1C40F'; mBg = 'rgba(241, 196, 15, 0.2)'; isPercentage = false;
    } else if (chartType === 'ir') {

```

```

        mLabel = 'Εσωτερική Αντίσταση (mΩ)'; mData = irData; mBorder = '#E74C3C'; mBg =
'rgba(231, 76, 60, 0.2)'; isPercentage = false;
    }

    // Κατασκευή του αναλυτικού γραφήματος πλήρους ανάλυσης
    modalChartInstance = new Chart(document.getElementById('modalCanvas').getContext('2d'), {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: mLabel,
                data: mData,
                borderColor: mBorder,
                backgroundColor: mBg,
                borderWidth: 3,
                fill: true,
                tension: 0.4
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            // Κλείδωμα κατακόρυφου άξονα [0, 100] για ποσοστιαία μεγέθη (SoC, SoH)
            scales: isPercentage ? { y: { min: 0, max: 100 } } : {}
        }
    });
}

// Κλείσιμο αναδυόμενου παραθύρου και αποδέσμευση γραφικών πόρων
function closeModal() {
    document.getElementById('chartModal').style.display = 'none';
    if (modalChartInstance) {
        modalChartInstance.destroy();
    }
}

```

```
    }  
  }  
  
  // Αυτόματο κλείσιμο εάν ο χρήστης κάνει κλικ εκτός των ορίων του modal  
  window.onclick = function(event) {  
    if (event.target !== document.getElementById('chartModal')) {  
      closeModal();  
    }  
  }  
}  
</script>  
{% endblock %}
```