ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΛΗΨΗΣ ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΤΟ ΖΩΝΤΑΝΟ ΕΡΓΑΣΤΗΡΙΟ ΤΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Κάργας Δ. Δημήτριος Α.Μ.: 3924

Επιβλέπων: Χριστοφορίδης Γεώργιος, Καθηγητής
Συνεπιβλέπων: Φιλιππίδης Σταύρος, Υποψήφιος Διδάκτωρ

(Υπογραφή)

.....

Κάργας Δ. Δημήτριος

Ηλεκτρολόγος Μηχανικός Τ.Ε., Πανεπιστήμιο Δυτικής Μακεδονίας

© 2022 – All rights reserved

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται τη διαδικασία αυτοματοποίησης για την λήψη μετρήσεων από μετρητές του ζωντανού εργαστηρίου του Πανεπιστημίου Δυτικής Μακεδονίας που βρίσκεται στα Κοίλα Κοζάνης. Στο πρώτο κεφάλαιο παρουσιάζονται σύντομα πληροφορίες για το ζωντανό εργαστήριο, και αναλυτικά πληροφορίες για τον εξοπλισμό που έχει τοποθετηθεί. Έμφαση δίνεται στο τρόπο λήψης μετρήσεων από την κάθε συσκευή. Έπειτα, στο δεύτερο κεφάλαιο, παρουσιάζονται αναλυτικά οι τέσσερις αλγόριθμοι που αναπτύχθηκαν για τη λήψη μετρήσεων από 4 διαφορετικούς τύπους συσκευών. Οι κώδικες παρατίθενται στο παράρτημα της εργασίας. Τέλος, παρουσιάζονται συντόμως ενδεικτικά μετρήσεις που λήφθηκαν μέσω των προαναφερθέντων αλγορίθμων.

Λέξεις Κλειδιά: Ζωντανό εργαστήριο, python, λήψη μετρήσεων, αυτοματοποίηση

ABSTRACT

This thesis deals with the automation process for data acquision from meters of the living laboratory of the University of Western Macedonia located in Koila Kozani. In the first chapter, brief information about the living laboratory is presented, and detailed information about the installed equipment. Emphasis is given on how data is acquired from each device. Then, in the second chapter, the four algorithms developed for data acquision from 4 different types of devices are presented in detail. The codes are listed in the appendix of the thesis. Finally, illustrative measurements obtained through the aforementioned algorithms are presented shortly.

Keywords: Living lab, Python, Data Acquisition, automatization

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κύριο Χριστοφορίδη Γεώργιο, που μου έδωσε την δυνατότητα να εκπονήσω την διπλωματική αυτή, και κυρίως τον υποψήφιο διδάκτορα κύριο Φιλιππίδη Σταύρο, για τον διαρκή του αγώνα και ζήλο που έδειξε, έτσι ώστε να παραδώσω την παρούσα εργασία, ολοκληρωμένη και σωστή.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΕΣ

ΕΙΣΑΓΩΓΗ

Το ζωντανό εργαστήριο του Πανεπιστημίου Δυτικής Μακεδονίας (ΠΔΜ) δημιουργήθηκε στο κτίριο των φοιτητικών εστιών στα Κοίλα, μέσω χρηματοδοτήσεων από ερευνητικά έργα. Το ζωντανό εργαστήριο είναι ένα οικοσύστημα ανοικτής καινοτομίας, που ενσωματώνει τις διαδικασίες έρευνας και καινοτομίας σε μια πραγματική κοινότητα και περιβάλλον, επικεντρωμένη στους χρήστες, δηλαδή τους φοιτητές που κατοικούν στις εστίες. Σκοπός του είναι η έρευνα και ανάπτυξη για την επίλυση σύνθετων κοινωνικών αναγκών. Στο κτίριο των εστιών, έχει εγκατασταθεί ένα ολοκληρωμένο σύστημα παραγωγής ηλεκτρικής ενέργειας από φωτοβολταϊκά.

Ένα σύστημα αποθήκευσης ενέργειας με μπαταρίες, ένας φορτιστής ηλεκτρικών αυτοκινήτων, μετρητικά όργανα για τη μέτρηση ηλεκτρικών μεγεθών σε 150 έξυπνες πρίζες, μετεωρολογικών δεδομένων, θερμοκρασίας κτιρίου σε 18 σημεία κλπ, και ελεγχόμενες συσκευές, όπως διακόπτες για τον έλεγχο του φωτισμού, και ηλεκτρονικές διατάξεις για ελεγχόμενες προσομοιώσεις.

Μεταξύ αυτών, περιλαμβάνονται τρία ηλεκτρονικά φορτία εγκατεστημένα στις τρεις επιμέρους φάσεις του δικτύου των εστιών, για ελεγχόμενες προσομοιώσεις παράλληλα χρησιμοποιώντας και τον υπόλοιπο εξοπλισμό, όπως τους αντιστροφείς, τις μπαταρίες και τον φορτιστή αυτοκινήτου.

Στα πλαίσια του ζωντανού εργαστηρίου έχουν δοθεί δεκάδες πτυχιακές εργασίας, και έχουν γίνει εκπαιδευτικές εκδρομές και παρουσιάσεις σε φοιτητές του τμήματος ΗΜΜΥ. Η θεματολογία των εργασιών αφορά τη λήψη και τον έλεγχο μετρήσεων, τον έλεγχο των συσκευών, τη σχεδίαση και εκτέλεση πειραμάτων.

1 ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΖΩΝΤΑΝΟΥ ΕΡΓΑΣΤΗΡΙΟΥ

Το Πανεπιστήμιο Δυτικής Μακεδονίας διαθέτει σήμερα, στην Κοζάνη, 145 κλίνες στα Κοίλα στην φοιτητική εστία που δημιουργήθηκε το 1986 στο πρώην Τεχνολογικό Εκπαιδευτικό Ίδρυμα (ΤΕΙ) Δυτικής Μακεδονίας. Ο χώρος των εστιών απεικονίζεται στο κάτω μέρος της παρακάτω εικόνας.

Figure 1: Αεροφωτογραφία Πανεπιστημίου Δυτικής Μακεδονίας στα Κοίλα Κοζάνης

Οι εστίες χωρίζονται σε τρεις (3) πτέρυγες οι οποίες διαχωρίζονται χρησιμοποιώντας χρώματα. Χωρίζονται σε κόκκινα, μπλε και πράσινα δωμάτια. Το κάθε δωμάτιο ονομάζεται βάσει ενός αριθμού και του χρώματος της πτέρυγάς του, για παράδειγμα, κόκκινο 113.

Μέσω χρηματοδοτήσεως από τα ερευνητικά έργα BERLIN, PV-Estia και MOST, ο χώρος των εστιών μετατράπηκε σε ένα ζωντανό εργαστήριο, στο οποίο λαμβάνονται μετρήσεις από όλα τα ηλεκτρικά μεγέθη του εργαστήριου. Επιπροσθέτως, ελέγχονται ορισμένα φορτία και πραγματοποιείται έρευνα για τα έξυπνα δίκτυα ηλεκτρικής ενέργειας, χρησιμοποιώντας ελεγχόμενα φορτία, το φωτοβολταϊκό σύστημα, τους μετατροπείς, τις έξυπνες συσκευές και το κομμάτι της δικτύωσης των συσκευών. Το εργαστήριο εμπλουτίζεται διαρκώς τόσο ως προς τις τεχνικές υποδομές του, όσο και ως προς το λογισμικό του. Επιπρόσθετα, στο ζωντανό εργαστήριο συλλέγονται και αξιοποιούνται δεδομένα από εγκαταστάσεις που έχουν γίνει και εκτός των εστιών, όπως στο χωριό Κοιλάδα.

Σε αυτό το κεφάλαιο παρουσιάζονται αρχικά οι συσκευές που έχουν τοποθετηθεί στο ζωντανό εργαστήριο, και στις υπόλοιπες τοποθεσίες και ο τρόπος που συνδέονται μεταξύ τους.

1.1 Εξοπλισμός

Ο πίνακας 1 παρουσιάζει τον εξοπλισμό που παράγει μετρήσεις οι οποίες αποθηκεύονται στη βάση δεδομένων του ζωντανού εργαστηρίου.

Table 1: Συσκευές εκ των οποίων λαμβάνονται μετρήσεις στο ζωντανό εργαστήριο

Α/Α	**Είδος**	**Τοποθέτηση**	**Λήψη μετρήσεων από τη συσκευή (την 28/11/22)**	**Τρόπος σύνδεσης για λήψη μετρήσεων**
1	Φωτοβολταϊκά	Κοίλα	Ναι	Μέσω modbus TCP στον inverter
2	-//-	Κοιλάδα	Ναι	Μέσω modbus TCP στον inverter
3	Inverter	Κοίλα	Ναι	Μέσω modbus TCP στον inverter
4	-//-	Κοιλάδα	Ναι	Μέσω modbus TCP στον inverter
5	Μπαταρίες	Κοίλα	Ναι	Μέσω modbus TCP στον inverter
6	-//-	Κοιλάδα	Ναι	Μέσω modbus TCP στον inverter
7	Ελεγκτής Siemens	Κοίλα	Όχι	-
8	-//-	Κοιλάδα	Όχι	-
9	Μετρητικά Smart Meter	Κοίλα	Όχι	-
10	-//-	Κοιλάδα	Όχι	-
11	Μετεωρολογικός σταθμός	Κοίλα	Ναι	Μέσω λογισμικού σε Η/Υ στο εσωτερικό δίκτυο
12	-//-	Κοιλάδα	Ναι	Μέσω λογισμικού σε Η/Υ στο εσωτερικό δίκτυο
13	Φορτιστής Αυτοκινήτου	Κοίλα	Ναι	Μέσω ιστοσελίδας του φορτιστή στο εσωτερικό δίκτυο
14	Μετρητικά DimSwitch Ham	Κοίλα	Ναι	Μέσω wi-fi και αποθήκευση σε server της εταιρείας
15	-//-	Κοιλάδα	Ναι	Μέσω wi-fi και αποθήκευση σε server της εταιρείας
16	Μετρητικά Smart Plugs	Κοίλα	Ναι	Μέσω wi-fi και αποθήκευση σε server της εταιρείας

17	17 Raspberry Pi	Κοίλα	Ναι	Μέσω εσωτερικού wi-fi

1. Μικροελεγκτής χαμηλού κόστους ανοιχτού κώδικα Raspberry Pi

Το Raspberry Pi είναι ένας υπολογιστής σε μέγεθος πιστωτικής κάρτας που δημιουργήθηκε από τη μη κερδοσκοπική Raspberry Pi Foundation στο Ηνωμένο Βασίλειο. Η εικόνα 1 απεικονίζει ένα από τα πρώτα μοντέλα του μικροελεγκτή. Ο μικροελεγκτής Pi είναι ένα διαδεδομένο εργαλείο για την δημιουργία εφαρμογών στους αυτοματισμούς λόγω των δυνατοτήτων του σε εισόδους/εξόδους, της επεξεργαστικής ισχύος του, και ειδικά του λογισμικού του. Το Raspberry Pi διαθέτει ακίδες GPIO κατά μήκος της άνω άκρης της πλακέτας. GPIO (General Purpose Input Output) σημαίνει είσοδος/έξοδος γενικού σκοπού. Αυτές οι ακίδες αποτελούν μια φυσική διεπαφή μεταξύ του Raspberry Pi και του εξωτερικού κόσμου, και λειτουργούν αποκλειστικά σε 3.3 V τάση.

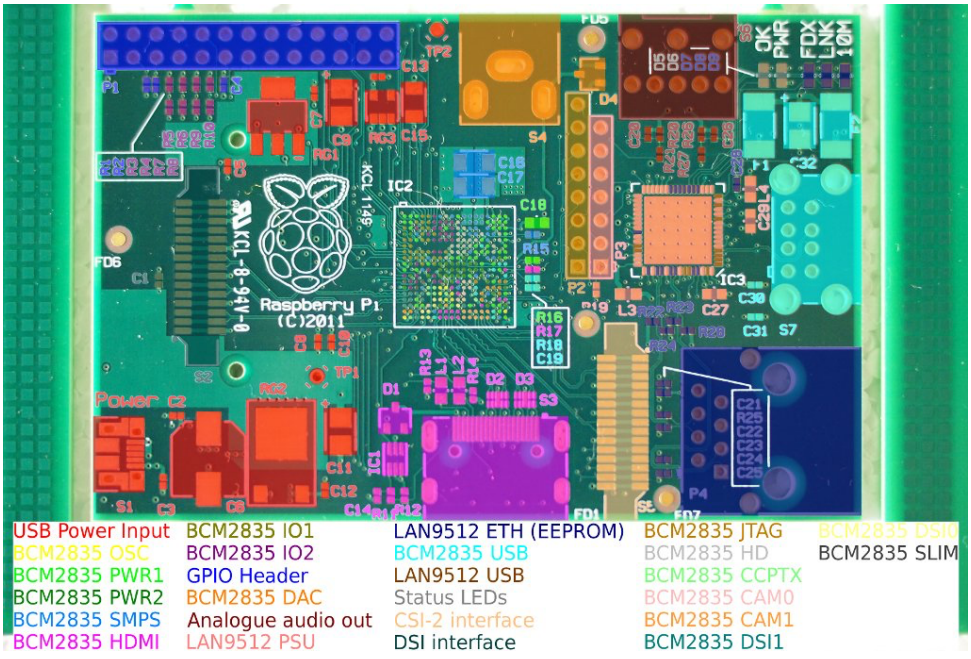

Figure 2: Δομή μικροελεγκτή Raspberry Pi 1

Το λειτουργικό σύστημα που χρησιμοποιεί ονομάζεται Raspberry Pi OS, είναι ένα ελεύθερο λειτουργικό σύστημα που βασίζεται στην διανομή του Debian GNU/Linux και είναι βελτιστοποιημένο για τα ηλεκτρονικά του Raspberry Pi (αρχιτεκτονική επεξεργαστή ARM).
Η χρήση του λειτουργικού συστήματος Raspberry Pi OS είναι ένα ακόμα από τα πλεονεκτήματα του μικροελεγκτή γιατί μας δίνει την δυνατότητα να χρησιμοποιήσουμε σχεδόν όλες τις εφαρμογές, σε επίπεδο λογισμικού, που προσφέρει η διανομή του Debian Linux, όπως ή γλώσσες προγραμματισμού C, Python κλπ. για την δημιουργία εφαρμογών αυτοματισμού.

Στην παρούσα εγκατάσταση, τα PI χρησιμοποιούνται για τη μέτρηση της θερμοκρασίας και της υγρασίας, σε 17 σημεία στο εσωτερικό των εστιών. Η μέτρηση γίνεται χρησιμοποιώντας έναν αισθητήρα DHT 11. Η εικόνα 2 απεικονίζει έναν αισθητήρα DHT 11 (αριστερά) και τον τρόπο συνδεσμολογίας του στο Raspberry PI (δεξιά).

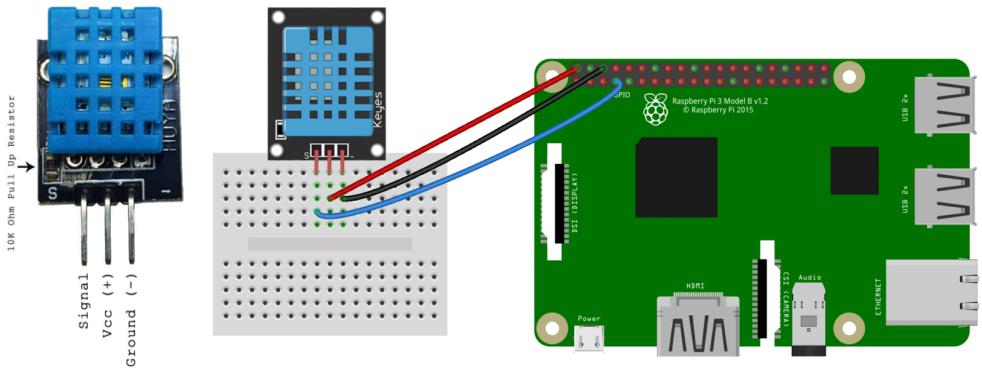

Figure 3: Φωτογραφία αισθητήρα DHT11 (αριστερά), συνδεσμολογία με RPi (δεξιά)

2. *Σταθμός φόρτισης Legrand Premium Green'up Station 22kW*

Στον εξωτερικό χώρο του ζωντανού εργαστηρίου έχει εγκατασταθεί και έχει τεθεί σε λειτουργία σταθμός φόρτισης ηλεκτρικών αυτοκινήτων ισχύος 22kW (εικόνα 2). Ο σταθμός φόρτισης είναι ενσωματωμένος στο δίκτυο ισχύος του ζωντανού εργαστηρίου και αποτελεί ένα από τα μεγαλύτερα ενεργά στοιχεία σε κατανάλωση ενέργειας. Ο σταθμός φόρτισης έχει την δυνατότητα για ταυτόχρονη φόρτιση 2 οχημάτων.

Figure 4: Σταθμός φόρτισης Legrand Premium Station 22kW

Ο σταθμός δημιουργεί δικό του web interface για τη διαχείριση και λήψη δεδομένων από αυτόν. Στην επόμενη εικόνα παρουσιάζεται ένα στιγμιότυπο του interface, το οποίο απεικονίζει τη λήψη δεδομένων σε αρχείο excel.

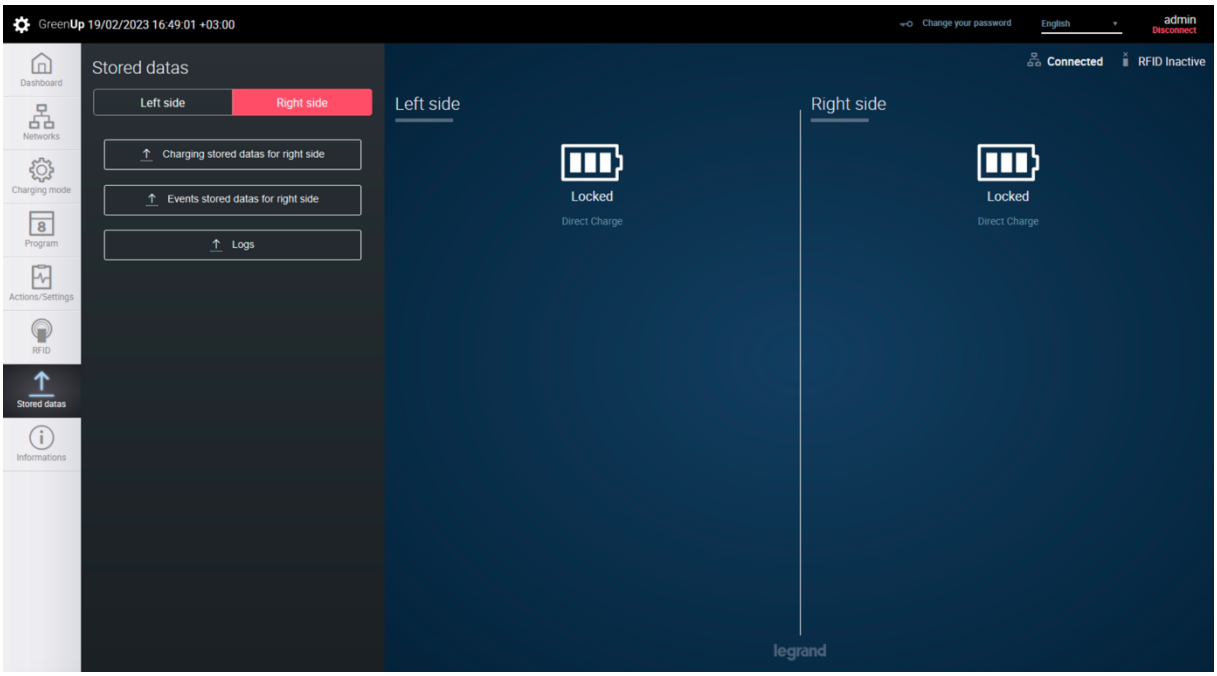

Figure 5: Στιγμιότυπο interface φορτιστή αυτοκινήτων

Ο φορτιστής θα συνδεθεί μέσω μετατροπέα πρωτοκόλλου OCPP σε modbus στο Gridvis, το οποίο εξηγείται παρακάτω.

3. Υβριδικός μετατροπέας FRONIUS GEN24

Για τη σύνδεση των φωτοβολταϊκών που εγκαταστάθηκαν με το δίκτυο, χρησιμοποιούνται Inverter της εταιρείας Fronius, τύπου GEN24, οι οποίοι έχουν και τη δυνατότητα αποθήκευσης της παραγόμενης ενέργειας σε συστοιχίες μπαταριών.

Figure 6: Υβριδικός μετατροπέας Fronius GEN24

Σημαντικό χαρακτηριστικό του υβριδικού μετατροπέα Fronius GEN24 είναι η μεγάλη ισχύς με την οποία μπορεί να φορτίσει αλλά και να εκφορτίσει τη συστοιχία μπαταριών που είναι συνδεδεμένος, με ισχύ έως 9kW.

Ο Fronius GEN24 διαθέτει δικό του ενσωματωμένο API, το οποίο ονομάζεται Solar API. Το Solar API είναι ένα ανοικτό REST API που επιτρέπει σε εφαρμογές να διαβάζουν δεδομένα

απευθείας από τον αντιστροφέα μέσω του δικτύου. Ο αντιστροφέας (ή το Datamanager 2.0) χρησιμεύει ως κεντρική μονάδα επικοινωνίας και καθιστά δεδομένα όπως ισχύς, τάση, ρεύμα κ.λπ διαθέσιμα. Μπορούν επίσης να ζητηθούν δεδομένα από εξαρτήματα που είναι συνδεδεμένα στον αντιστροφέα, όπως Smart Meter, Ohmpilot, String Control κ.λπ. Όλα τα δεδομένα επιστρέφονται ως αντικείμενα JSON.

4. Μπαταρίες

Οι συστοιχίες μπαταριών συνδέονται είναι τύπου BYD B-box HV 10.2, με ονομαστική χωρητικότητα 10.24 kWh το κάθε ένα από τα δυο συστήματα. Η εικόνα 7 απεικονίζει μία συστοιχία μπαταριών, και ο πίνακας 2 τα χαρακτηριστικά αυτού.

Η λήψη των μετρήσεων από τις μπαταρίες πραγματοποιείται μέσω του του inverter, καθώς αυτές συνδέονται απευθείας και επικοινωνούν με αυτόν.

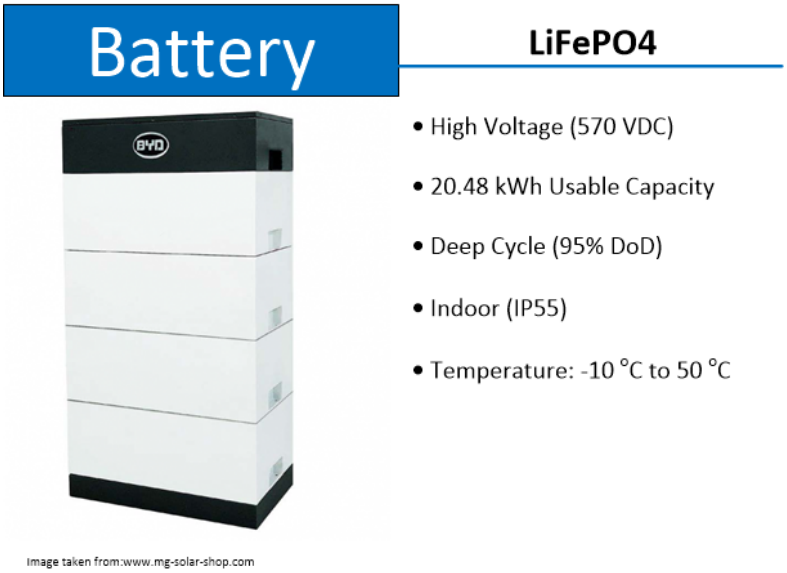

Figure 7: Μπαταρία τύπου BYD B-box HV10.2

Table 2: Χαρακτηριστικά μπαταριών τύπου BYD B-box HV10.2

Battery unit (x2)	**BYD B-box HV 10.2**
Technology	Lithium-Iron Phosphate ($LiFEPO_4$)
Energy capacity (nominal/usable)	10.24/10.24 kWh
Maximum charge/discharge power	10.24 kW
Relative humidity range	----
Depth of Discharge (DoD)	95%
Communication with the inverter	RS485
IP rate	IP55
Operating Temperature Range	-10 to 50 °C
Installation	Indoor

5. Μετρητικά HAM Dinswitch

Το HAM DinSwitch είναι ένας διακόπτης και μετρητής ενέργειας σχεδιασμένος για τοποθέτηση σε Ηλεκτρολογικούς Πίνακες. Η παρακάτω εικόνα απεικονίζει τη συσκευή HAM Dinswitch.

Figure 8: Φωτογραφία μετρητή HAM Dinswitch

Το dinswitch λειτουργεί με 230VAC, 50Hz τάση δικτύου, και έχει τη δυνατότητα να ελέγχει φορτία μέχρι 2A, φωτισμό ή ρελέ για οτιδήποτε άλλο (π.χ. boiler, φούρνος). Λαμβάνει μετρήσεις κατανάλωσης χρησιμοποιώντας την παλμική έξοδο S0 (DIN 43864) από μετρητές κιλοβατώρας που έχουν εγκατασταθεί στον ίδιο πίνακα. Λειτουργεί μέσω δικτύου Wi-fi.

Κάθε συσκευή έχει έναν αύξοντα αριθμό της μορφής F:S (π.χ. 12:34). Το μέρος F του σειριακού αριθμού είναι η οικογένεια προϊόντος και είναι κοινό σε παρόμοιες συσκευές. Για παράδειγμα, όλες οι συσκευές DinSwitch έχουν σειριακό αριθμό με τη μορφή 12:S, ενώ όλες οι συσκευές Plug έχουν 14:S κ.ο.κ. Το S είναι ένας αύξων αριθμός, μοναδικός για κάθε συσκευή εντός της ίδιας οικογένειας. Ο σειριακός αριθμός μιας συσκευής είναι το αναγνωριστικό που χρησιμοποιείται από τον διακομιστή, το UI και τα τελικά σημεία API. Παραδείγματα σειριακών αριθμών είναι τα εξής: 12:4, 12:7, 14:20, 8:4.

Για κάθε συσκευή δημιουργούνται δύο τυχαία κλειδιά. Το ένα είναι το device_key και είναι ο κωδικός πρόσβασης WPA του σημείου πρόσβασης (AP) και χρησιμοποιείται επίσης για την καταχώρηση μιας συσκευής. Το δεύτερο κλειδί είναι μια μακρά ακολουθία που χρησιμοποιεί ο διακομιστής για να πιστοποιήσει τη σύνδεση μιας συσκευής και η συσκευή για να πιστοποιήσει τον διακομιστή. Όλες οι επικοινωνίες και οι αλληλεπιδράσεις με τις συσκευές πραγματοποιούνται μέσω του διακομιστή από προεπιλογή. Υπάρχει η δυνατότητα αποστολής αιτημάτων HTTP με εντολές συσκευής απευθείας στις συσκευές στη διεύθυνση IP τους, μόλις συνδεθούν σε ένα τοπικό δίκτυο, αλλά είναι απενεργοποιημένη από προεπιλογή για λόγους

ασφαλείας. Ο διακομιστής πιστοποιεί τις συσκευές, πιστοποιεί τους πελάτες και διαβιβάζει εντολές και δεδομένα μεταξύ των δύο. Ο διακομιστής χειρίζεται, μεταξύ άλλων, την καταγραφή δεδομένων και την αξιολόγηση κανόνων.

Ο συνδεδεμένος χρήστης καθορίζεται από τον διακομιστή, χρησιμοποιώντας ένα παρεχόμενο api_key ή ένα access_token μέσω των παραμέτρων COOKIE, POST ή GET. Σε περίπτωση που το access_token ή το api_key παρέχονται πολλές φορές, το api_key έχει προτεραιότητα έναντι του access_token και στη συνέχεια, οι παράμετροι GET έχουν την υψηλότερη προτεραιότητα, οι παράμετροι POST είναι οι επόμενες και τελευταίες είναι οι παράμετροι COOKIE.

Ένα διακριτικό πρόσβασης (access token) δημιουργείται με τον τρόπο OAuth2 όταν ένας χρήστης συνδέεται χρησιμοποιώντας το email και τον κωδικό πρόσβασής του. Πολλαπλά access_tokens μπορούν να είναι ενεργά μέχρι ένα ορισμένο όριο. Όταν ξεπεραστεί αυτό το όριο, τα παλαιότερα access_tokens αντικαθίστανται και ακυρώνονται. Τα access tokens έχουν επίσης ημερομηνία λήξης. Αυτό σημαίνει ότι τα access tokens είναι προσωρινά.

Η συλλογή δεδομένων γίνεται αποκλειστικά σε server της εταιρείας HAM. Η εταιρεία έπειτα επιτρέπει στους χρήστες να προβάλλουν και να κατεβάσουν τα δεδομένα τους μέσω της πλατφόρμας https://hamsystems.eu. Επιπρόσθετα, παρέχει πρόσβαση σε δεδομένα ιστορικού από την Python. Ο ευκολότερος τρόπος είναι να χρησιμοποιηθεί το πακέτο pip που είναι διαθέσιμο στο pypi, με όνομα hamapi.

Ακολουθεί ένα παράδειγμα χρήσης του πακέτου hamapi για τη λήψη δεδομένων ιστορικού:

```
1 import hamapi
2
3 # set you API key here
4 ham = hamapi.hamapi(api_key='', server='hamsystems.eu')
5 # unix timestamp of the start of data
6 start = math.floor(time.time()-7*86400)
7 # unix timestamp of the end of data
8 end = math.floor(time.time()-86400)
9 # step to smooth/average the data in seconds
10 step = 2*60
11 # get datalog data for device with serialno 12:204, from start to end with step
12 data = ham.get_datalog_data("12:204", start, end, step)
13 print(data)
```

6. Μετρητικά HAM Smart Plugs

Τα HAM Plug είναι συσκευές που τοποθετούνται στο εσωτερικό ενός κουτιού πρίζας. Επιτρέπει τη λήψη μετρήσεων σε πραγματικό χρόνο για 5 μετρικά: ενεργή ισχύ, άεργο ισχύ, κατανάλωση ρεύματος, διαφορά δυναμικού και καταναλισκόμενη ενέργεια. Επιπρόσθετα,

επιτρέπει τον έλεγχο (on/off) οποιασδήποτε ηλεκτρικής συσκευής που συνδέεται στην πρίζα έως 16A. Η παρακάτω εικόνα απεικονίζει ένα HAM Smart Plug.

Figure 9: HAM Smart plug

Λειτουργεί με WiFi, και η λήψη δεδομένων γίνεται με ακριβώς τον ίδιο τρόπο όπως οι συσκευές HAM Dinswitch. όμως απαιτείται επιπρόσθετα να οριστούν τα 5 μεγέθη που μετράει η πρίζα, διότι μοιράζονται το ίδιο ID.

7. Μετεωρολογικός Σταθμός – καταγραφικό Stylitis-10

Ο μετεωρολογικός σταθμός αποτελείται από 5 αυτόνομους αισθητήρες, οι οποίοι συνδέονται σε ένα καταγραφικό σύστημα τύπου Stylitis-10. Το καταγραφικό επικοινωνεί με τους επιμέρους αισθητήρες χρησιμοποιώντας αναλογικά σήματα, και δίνει τη δυνατότητα επικοινωνίας μαζί του μέσω του λογισμικού OPTON. Οι επιμέρους αισθητήρες είναι οι εξής:

1. Πυρανόμετρο Apogee SP-100
2. Θερμόμετρο/υγρόμετρο εξωτερικού χώρου Thygro
3. Θερμόμετρο/υγρόμετρο εσωτερικού χώρου Thygro
4. Θερμόμετρο επιφάνειας Φ/Β πάνελ PT100
5. Ανεμόμετρο A75-104 Sine wave anemometer

Το καταγραφικό Stylitis-10 διαθέτει 8 κανάλια πολλαπλών λειτουργιών, τα οποία μπορούν να χρησιμοποιηθούν ως: είσοδοι μέτρησης (1-2 μετρητές και 6-7 αναλογικά κανάλια - 7 ψηφιακές είσοδοι), ή έξοδοι ελέγχου (αλλάζουν είτε χειροκίνητα είτε σε συνάρτηση με τις τιμές των εισόδων). Δειγματοληψία γίνεται μία φορά ανά δευτερόλεπτο.

Το λογισμικό OPTON είναι ελεύθερο και έχει εγκατασταθεί σε έναν υπολογιστή που λειτουργεί στο εσωτερικό τοπικό δίκτυο των εστιών. Τα αρχεία δεδομένων μπορούν να ληφθούν χειροκίνητα, μέσω του OPTON, για οποιαδήποτε σύνδεση του σταθμού.

Ωστόσο, η λήψη δεδομένων, καθώς και άλλες ενέργειες, μπορούν να πραγματοποιούνται αυτόματα, σε τακτική βάση, μέσω του πρόσθετου δωρεάν λογισμικού AutoConnect.

Εν προκειμένω, το λογισμικό AutoConnect χρησιμοποιείται για το κατέβασμα των μετρήσεων σε φάκελο στον τοπικό υπολογιστή, και ένα νέο σκριπτ αυτοματοποίησης απαιτείται για την επεξεργασία και αποθήκευση δεδομένων.

8. Λήψη μετρήσεων από μετρητές Janitza και Inverter Kostal

Οι μετρητές Janitza, καθώς και όσες συσκευές λειτουργούν με πρωτόκολλο Modbus over ethernet, συνδέονται στο λογισμικό Gridvis, και αποθηκεύουν τις μετρήσεις τους στη βάση δεδομένων που είναι εγκατεστημένη στον υπολογιστή όπου λειτουργεί το λογισμικό. Η πρόσβαση στα δεδομένα είναι δυνατή μέσω του γραφικού περιβάλλοντος του λογισμικού, ή μέσω του REST API που δημιουργεί το λογισμικό. Για την αυτοματοποίηση λήψης μετρήσεων, θα χρησιμοποιηθεί το API του λογισμικού. Τα API REST του SERVICE/WEB παρέχει πρόσβαση σε πόρους (οντότητες δεδομένων) μέσω συνδέσμων URL. Το SERVICE/WEB REST API χρησιμοποιεί XML, JSON και JSONP ως μορφή επικοινωνίας, καθώς και τυπικές μεθόδους HTTP όπως GET, PUT, POST και DELETE. Χρησιμοποιώντας το REST API, ο κώδικας λήψης μετρήσεων θα κάνει μια αίτηση HTTP και θα αναλύει την απόκριση.
Τα URI για τους πόρους του REST API του SERVICE/WEB έχουν την ακόλουθη δομή:

http://host:port/rest/api-version/resource-name

Υπάρχει ένα έγγραφο WADL που περιέχει την τεκμηρίωση για κάθε πόρο στο SERVICE/WEB REST API.
Το API έχει τη δυνατότητα να παράγει XML, JSON και JSONP. Ο τύπος περιεχομένου καθορίζεται από την τιμή της επικεφαλίδας HTTP "Accept". Ωστόσο, ειδικά για τις αιτήσεις AJAX από το πρόγραμμα περιήγησης, δεν είναι πάντα δυνατό να οριστούν οι τιμές των επικεφαλίδων. Σε αυτή την περίπτωση, ο επιθυμητός τύπος περιεχομένου μπορεί να προστεθεί στο url, αλλά πρέπει να οριστεί πριν από οποιεσδήποτε παραμέτρους ερωτήματος.
Παράδειγματα:

Auto: http://localhost:8080/1/projects/

XML: http://localhost:8080/1/projects/.xml

JSON: http://localhost:8080/1/projects/.json

Auto: http://localhost:8080/1/projects/?someParam=value

XML: http://localhost:8080/1/projects/.xml?someParam=value

JSON: http://localhost:8080/1/projects/.json?someParam=value

Η πρόσβαση στα API REST από διαφορετικό τομέα μέσω AJAX παραβιάζει την ίδια πολιτική προέλευσης. Επομένως, πρέπει να χρησιμοποιείται η τεχνική JSONP. Για να ενεργοποιήσετε

το JSONP, πρέπει να επιλεγεί ο τύπος αρχείου json και να δοθεί τη παράμετρος "callback" εντός της συνάρτησης επανάκλησης.

1.2 Συνδεσμολογία συσκευών

Ο πίνακας 3 παρουσιάζει τον τρόπο που συνδέονται οι συσκευές από τις οποίες λαμβάνονται μετρήσεις, ενώ η εικόνα 10 παρουσιάζει τον τρόπο που συνδέονται οι συσκευές με modbus σε μετρητές Janitza. Εν προκειμένω, παρουσιάζεται η σύνδεση των ινβέρτερ ως παράδειγμα.

A/A	Είδος	Τρόπος σύνδεσης
1	Fronius Inverter	Σύνδεση ethernet στο δίκτυο SMART και λήψη μέσω του API
2	Kostal Inverter	Σύνδεση modbus σε μετρητή Janitza και έπειτα στο gridvis
3	Μπαταρίες	Σύνδεση modbus σε μετρητή Janitza και έπειτα στο gridvis
4	Μετεωρολογικός σταθμός	Σύνδεση modbus σε μετρητή Janitza και έπειτα στο gridvis
5	Φορτιστής Αυτοκινήτου	Μετατροπέας OCPP σε Modbus και σύνδεση στο gridvis
6	Μετρητικά DimSwitch Ham	Μέσω σύνδεσης στο ασύρματο δίκτυο SMART
7	Μετρητικά Smart Plugs	Μέσω σύνδεσης στο ασύρματο δίκτυο SMART
8	17 Raspberry Pi	Μέσω σύνδεσης στο ασύρματο δίκτυο SMART

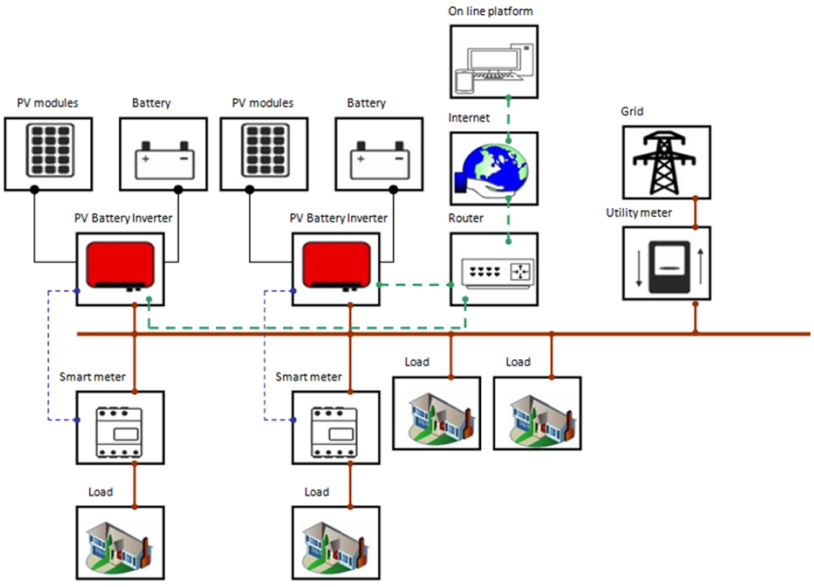

Figure 10: Συνδεσμολογία συσκευών που συνδέονται με modbus σε janitza (smart meter)

Ουσιαστικά οι συσκευές χωρίζονται άτυπα σε δύο τρόπους συνδεσμολογίας, οι μισές οι συνδέονται μέσω πρωτοκόλλου modbus σε μετρητές Janitza, ενώ οι υπόλοιπες συνδέονται απευθείας στο εσωτερικό δίκτυο που δημιουργήθηκε για το ζωντανό εργαστήριο.

Η εικόνα 11 απεικονίζει τα σημεία που βρίσκονται τοποθετημένες οι ασύρματες και ενσύρματες κεραίες στο ζωντανό εργαστήριο.

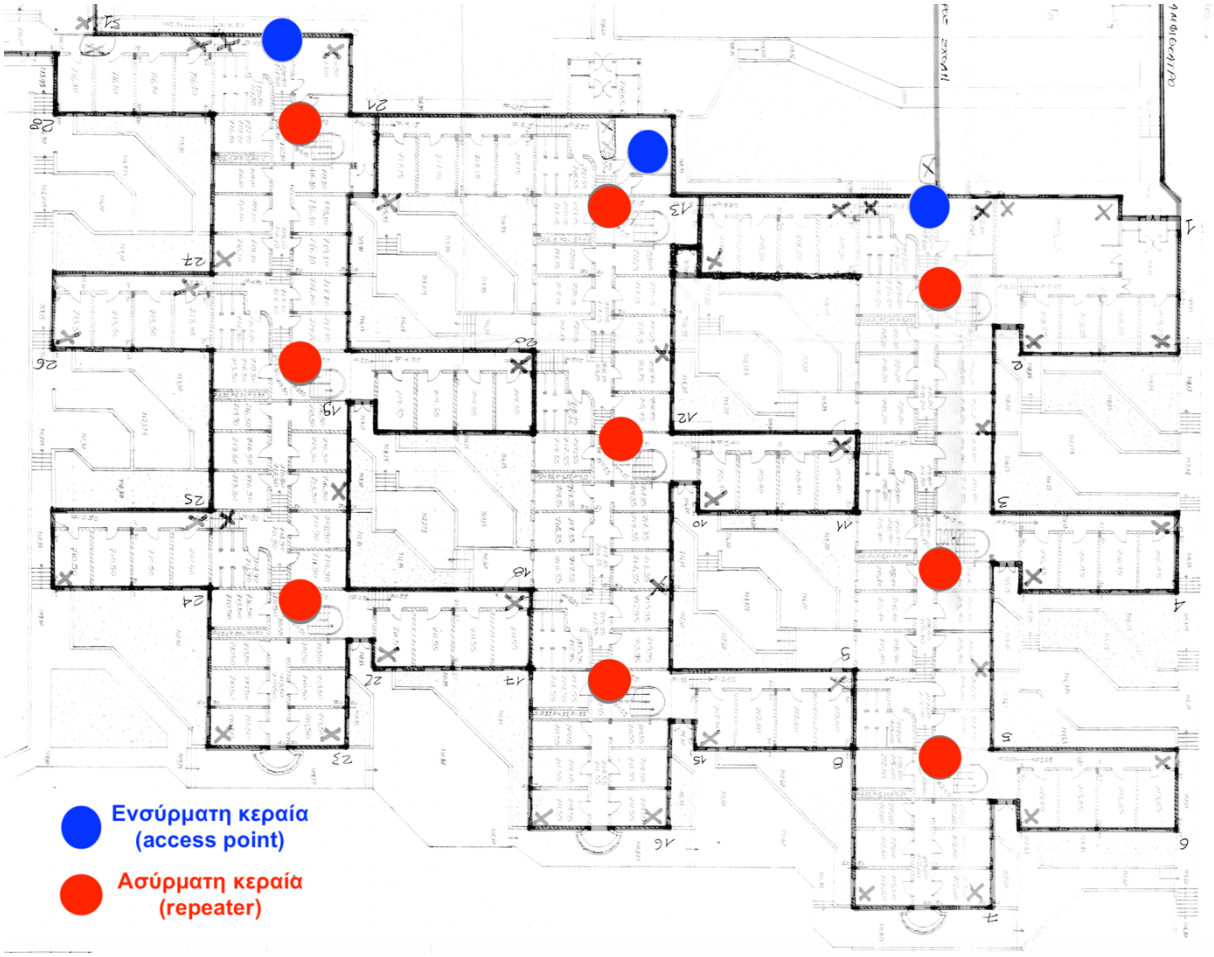

Figure 11: Κεραίες wi-fi living lab

2 ΛΗΨΗ ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΣΥΣΚΕΥΕΣ

Ο πίνακας 2 παρουσιάζει τον τρόπο που λαμβάνονται μετρήσεις για κάθε μία από τις συσκευές σε επίπεδο κώδικα.

Πίνακας 1: Τρόπος λήψης μετρήσεων ανά συσκευή του ζωντανού εργαστηρίου

Α/Α	Είδος	Τοποθέτηση	Τρόπος λήψης μετρήσεων
1	Φωτοβολταϊκά	Κοίλα	Μέσω του API του Inverter
2	-//-	Κοιλάδα	Μέσω του API του Inverter
3	Fronius Inverter	Κοίλα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που κατεβάζει από το API
4	-//-	Κοιλάδα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που κατεβάζει από το API
5	Kostal Inverter	Κοίλα	Μέσω σύνδεσης στο Gridvis ως συσκευές Modbus
6	Μπαταρίες	Κοίλα	Μέσω του API του Inverter
7	-//-	Κοιλάδα	Μέσω του API του Inverter
8	Ελεγκτής Siemens	Κοίλα	Όχι
9	-//-	Κοιλάδα	Όχι
10	Μετρητικά Smart Meter	Κοίλα	Όχι
11	Μετρητικά Smart Meter	Κοιλάδα	Όχι
12	Μετεωρολογικός σταθμός	Κοίλα	Μέσω λογισμικού της εταιρείας, αποθήκευση σε τοπικό Η/Υ και αυτοματοποίηση λήψης των αρχείων
13	-//-	Κοιλάδα	Μέσω λογισμικού της εταιρείας, αποθήκευση σε τοπικό Η/Υ και αυτοματοποίηση λήψης των αρχείων
14	Φορτιστής Αυτοκινήτου	Κοίλα	Ναι
15	Μετρητικά DimSwitch Ham	Κοίλα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που κατεβάζει από το API της εταιρείας των μετρητών
16	-//-	Κοιλάδα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που κατεβάζει από το API της εταιρείας των μετρητών
17	Μετρητικά Smart Plugs	Κοίλα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που κατεβάζει από το API της εταιρείας των μετρητών
18	17 Raspberry Pi	Κοίλα	Μέσω custom script στη γλώσσα προγραμματισμού Python, που εκτελείται σε κάθε ένα RPi και αυτοματοποίηση λήψης των αρχείων

Στην παρούσα εργασία, αναπτύχθηκαν οι κώδικες που αναφέρονται στον πίνακα για τη λήψη των μετρήσεων. Δηλαδή, αναπτύχθηκε ένας κώδικας που λαμβάνει δεδομένα από το API του Gridvis, ένας για τους inverter Fronius, ένας για τα Raspberry Pi, και ένας για τις συσκευές HAM. Σε αυτό το κεφάλαιο, παρουσιάζονται ξεχωριστά οι επιμέρους κώδικες.

2.1 Λήψη μετρήσεων από raspberry pi

Η λήψη μετρήσεων από τα raspberry pi γίνεται με τη χρήση κώδικα σε python ο οποίος χρησιμοποιεί τη βιβλιοθήκη της adafruit η οποία βρίσκεται διαθέσιμη ελεύθερα στο διαδίκτυο. Η βιβλιοθήκη χρησιμοποιεί τον ενσωματωμένο analog-to-digital converter του μετρητή ώστε να μετατρέπει το αναλογικό σε ψηφιακό σήμα, το οποίο μπορεί να χρησιμοποιηθεί από το raspberry pi. Η βιβλιοθήκη επίσης μετατρέπει το ψηφιακό σήμα σε κείμενο το οποίο μπορεί να χρησιμοποιήσει ο χρήστης ώστε να δει τη μέτρηση του αισθητήρα.

Το διάγραμμα ροής του κώδικα παρουσιάζεται στην επόμενη εικόνα. Ο κώδικας στην python βρίσκεται στο παράρτημα.

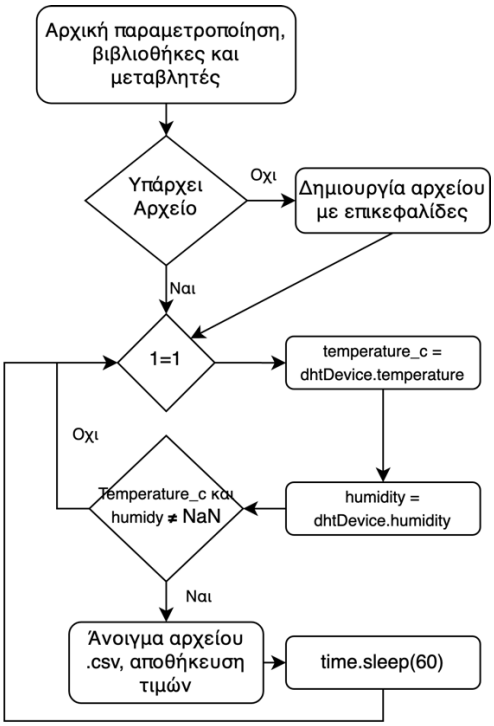

Figure 12: Διάγραμμα ροής κώδικα λήψης μετρήσεων από DHT11

Αρχικά εισάγονται οι απαραίτητες βιβλιοθήκες και αρχικοποιούνται μεταβλητές. Έπειτα, ελέγχεται εάν έχει δημιουργηθεί ένα αρχείο .csv ώστε ο κώδικας να το ανοίξει προκειμένου να αποθηκεύσει μετρήσεις. Εάν το αρχείο δεν υπάρχει, τότε δημιουργείται. Σε αυτό το σημείο ξεκινάει ένας ατέρμων βρόγχος, που εκτελείται όσο είναι ενεργό το Raspberry Pi. Σε αυτόν, λαμβάνονται γρήγορα πολλές τιμές της υγρασίας και τις θερμοκρασίας, και ελέγχεται εάν

υπάρχει μία τιμή που δεν είναι NaN. Εάν υπάρχει μία τιμή, τότε αυτή γράφεται στο αρχείο .csv και ο αλγόριθμος παγώνει για 60 δευτερόλεπτα, μετά τα οποία επιστρέφει στον ατέρμων βρόγχο.

2.2 Λήψη μετρήσεων από πολλαπλά HAM dinswitch ταυτόχρονα

Ο κώδικας που αναπτύχθηκε για τη λήψη μετρήσεων από HAM dinswitch, είναι κατασκευασμένος ώστε να μπορεί να λειτουργήσει και να αποθηκεύσει δεδομένα για πολλαπλά HAM dinswitch με μόνο μία εκτέλεση. Τούτο συμβαίνει δίνοντας την δυνατότητα να οριστούν πολλαπλοί σειριακοί αριθμοί, και επαναλαμβάνοντας τη διαδικασία για κάθε ένα από αυτούς. Αρχικά, ορίζεται η διεύθυνση του API της εταιρείας και ο απαραίτητος κωδικός. Έπειτα, ορίζεται η αρχική ημερομηνία από την οποία θα ληφθούν μετρήσεις, και το χρονικό βήμα, δηλαδή ανά πόσα λεπτά. Σε αυτό το σημείο αρχίζει ο κύριος βρόγχος επανάληψης του κώδικα, ο οποίος λαμβάνει το σειριακό αριθμό του μετρητή,

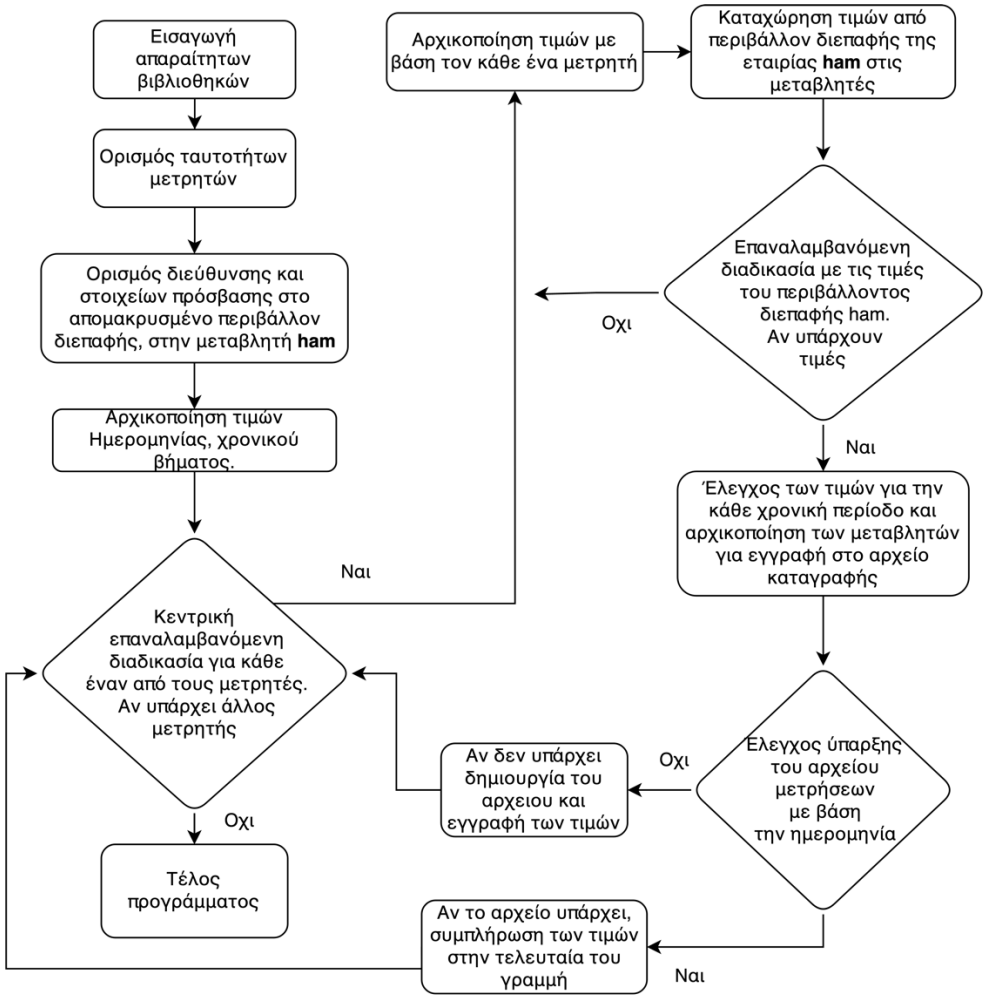

Figure 13: Διάγραμμα ροής λήψης μετρήσεων από συσκευές HAM

Όπως και στους άλλους αλγορίθμους, αρχικά εισάγονται οι απαραίτητες βιβλιοθήκες και αρχικοποιούνται κάποιες μεταβλητές. Αρχικοποιούνται επίσης τα στοιχεία που απαιτούνται για την πρόσβαση στο API της εταιρείας HAM, και οι τιμές της ημερομηνίας από της οποίας θα αρχίσει η λήψη μετρήσεων. Ο αλγόριθμος ξεκινά τον κύριο βρόγχο επανάληψης, που επαναλαμβάνεται για κάθε μία συσκευή HAM για την οποία έχει οριστεί ταυτότητα. Σε αυτό το βρόγχο, λαμβάνονται οι τιμές που υπάρχουν για τη συσκευή, και ελέγχεται εάν υπάρχει το αρχείο με βάση την ημερομηνία στο όνομα του αρχείου. Εάν δεν υπάρχει, τότε δημιουργείται με όνομα την ημερομηνία και ο αλγόριθμος συνεχίζει με την επόμενη συσκευή, ειδάλλως, εάν το αρχείο υπάρχει, οι μετρήσεις συμπληρώνονται στο τέλος των γραμμών του.

2.3 Λήψη μετρήσεων από μετρητές Janitza και συσκευές Modbus TCP

Η λήψη μετρήσεων από τους μετρητές janitza και των συσκευών που συνδέονται με πρωτόκολλο modbus over TCP γίνεται μέσω του λογισμικού gridvis το οποίο παρέχει API. Από το Gridvis λαμβάνονται 322 μετρικά, δηλαδή 322 τιμές από μεγέθη τα οποία αποθηκεύονται στο αρχείο .csv. Για κάθε ένα από τα μετρικά, απαιτείται ένας σύνδεσμος με την τοποθεσία του μετρικού στο σύστημα αρχείων του Gridvis. Έτσι, αρχικά δημιουργήθηκε ένα αρχείο εξέλ που αντιστοιχίζει τα διαφορετικά μετρικά με τις διευθύνσεις τους στο Gridvis. Για παράδειγμα, ο παρακάτω πίνακας παρουσιάζει μερικές από τις αντιστοιχίες:

Πίνακας 2: Παράδειγμα αντιστοίχισης μετρικών με σύνδεσμο στο Gridvis API

Μετρικό		Σύνδεσμος
Inverter III output-THD U L1	%	http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/18/hist/values/TotalHarmonicDisturbation_U/L1/900/?start=NAMED_Yesterday&end=NAMED_Today
Wing 1-1-Voltage L1	W	http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/13/hist/values/U_Effective/L1/900/?start=NAMED_Yesterday&end=NAMED_Today
Wing 1-1-Active power L1	W	http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/13/hist/values/PowerActive/L1/900/?start=NAMED_Yesterday&end=NAMED_Today

Ο κώδικας που αναπτύχθηκε λαμβάνει μετρήσεις για κάθε ένα από τα 322 μετρικά και τα αποθηκεύει σε αρχείο .csv με προεπιλεγμένο χρονικό βήμα, π.χ. 15 λεπτά. Στην επόμενη εικόνα παρουσιάζεται το διάγραμμα ροής του κώδικα:

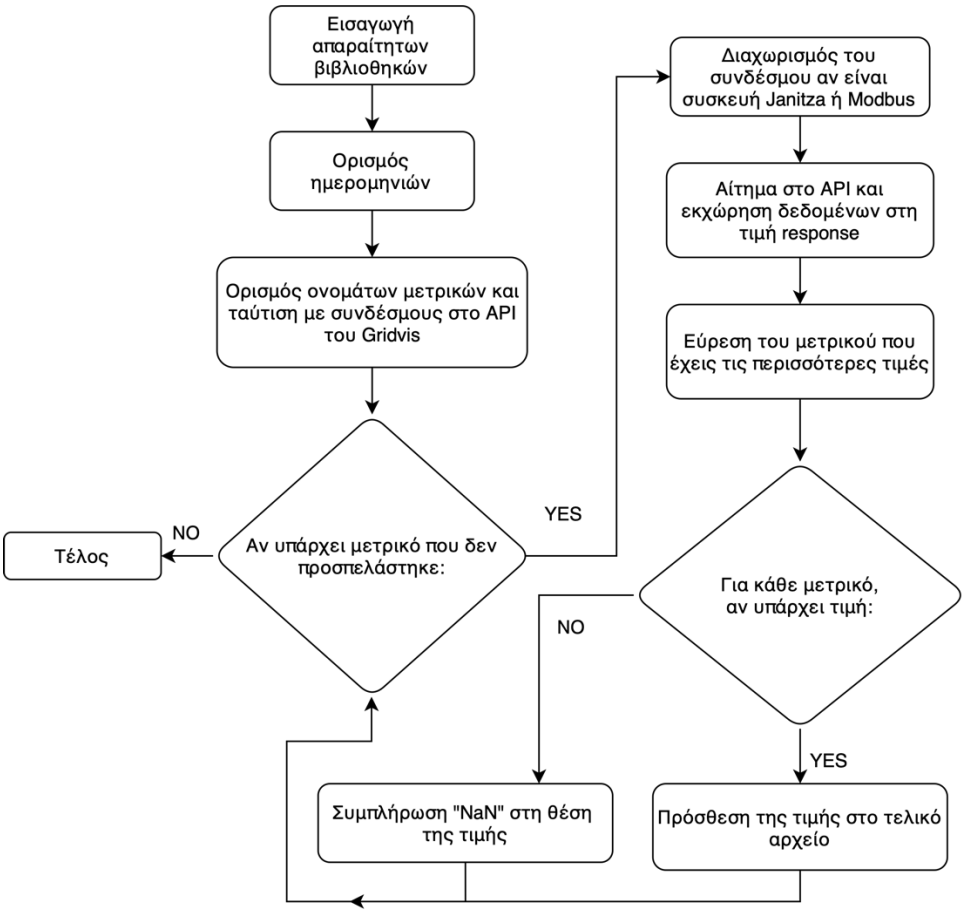

Figure 14: Διάγραμμα ροής κώδικα λήψης μετρήσεων από Gridvis

Ο φορτιστής αυτοκινήτου καθώς και οι μπαταρίες, συνδέονται ως συσκευές modbus στο Gridvis, συνεπώς οι μετρήσεις τους λαμβάνονται απευθείας μέσω αυτού.

3 ΕΝΔΕΙΚΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΜΕΤΡΗΣΕΩΝ

Χρησιμοποιώντας τους κώδικες που δημιουργήθηκαν στην εργασία, παρουσιάζονται ενδεικτικά τρία γραφήματα μετρήσεων ληφθεισών εκ των τριών διαφορετικών αλγορίθμων το κάθε ένα.

Η εικόνα 15 παρουσιάζει μετρήσεις που λήφθηκαν μέσω του αλγορίθμου για το inverter API:

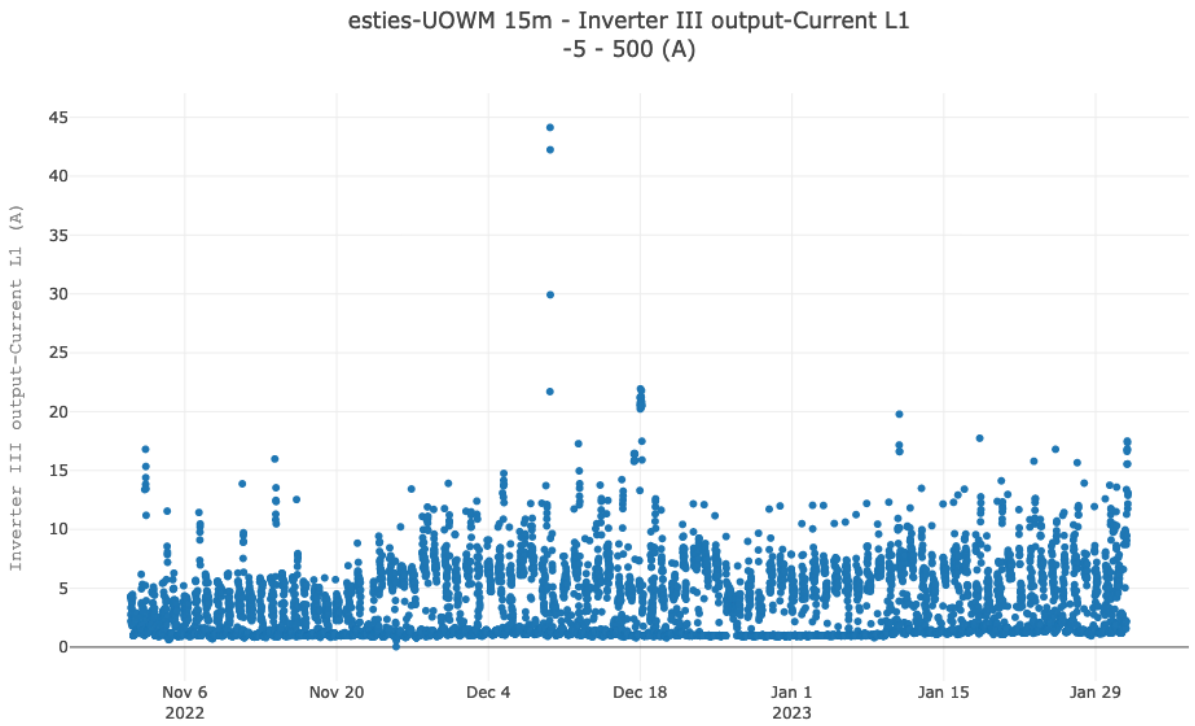

Figure 15: Μετρήσεις ρεύματος στην πρώτη φάση του Inverter

Η εικόνα 16 παρουσιάζει μετρήσεις που λήφθηκαν μέσω του αλγορίθμου για το Gridvis API:

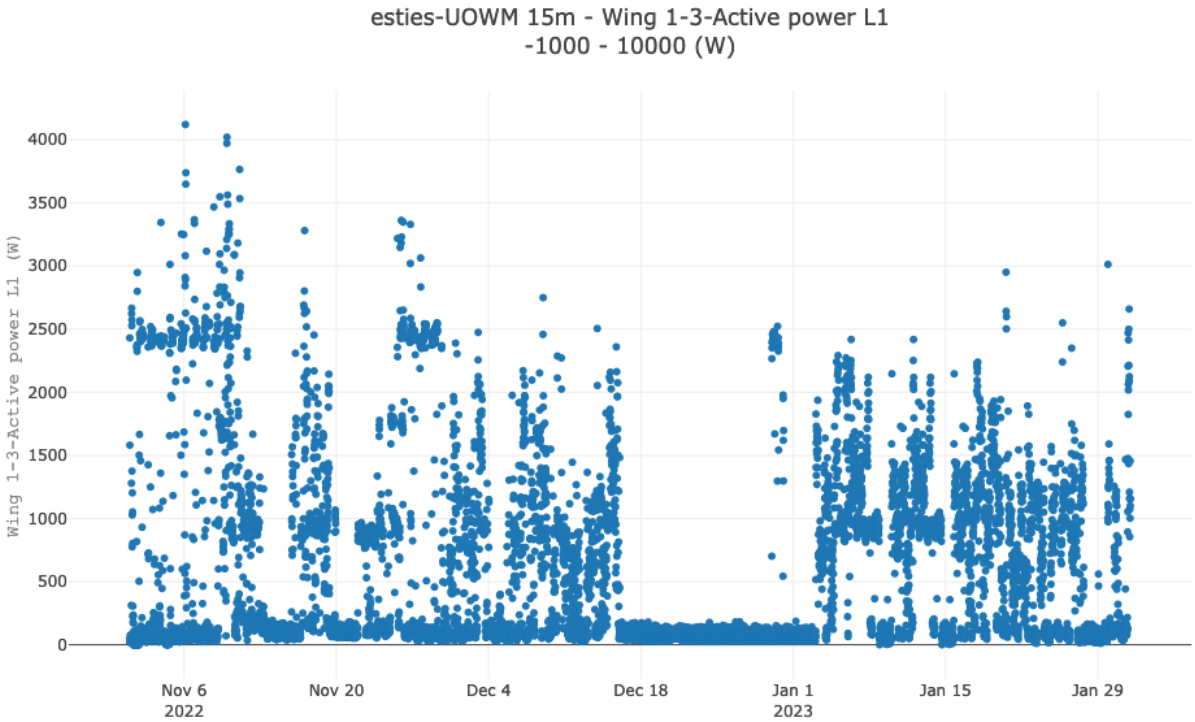

Figure 16: Μετρήσεις πραγματικής ισχύος στην πρώτη φάση της πρώτης πτέρυγας

Η εικόνα 18 παρουσιάζει μετρήσεις που λήφθηκαν μέσω του αλγορίθμου για τους prosumer:

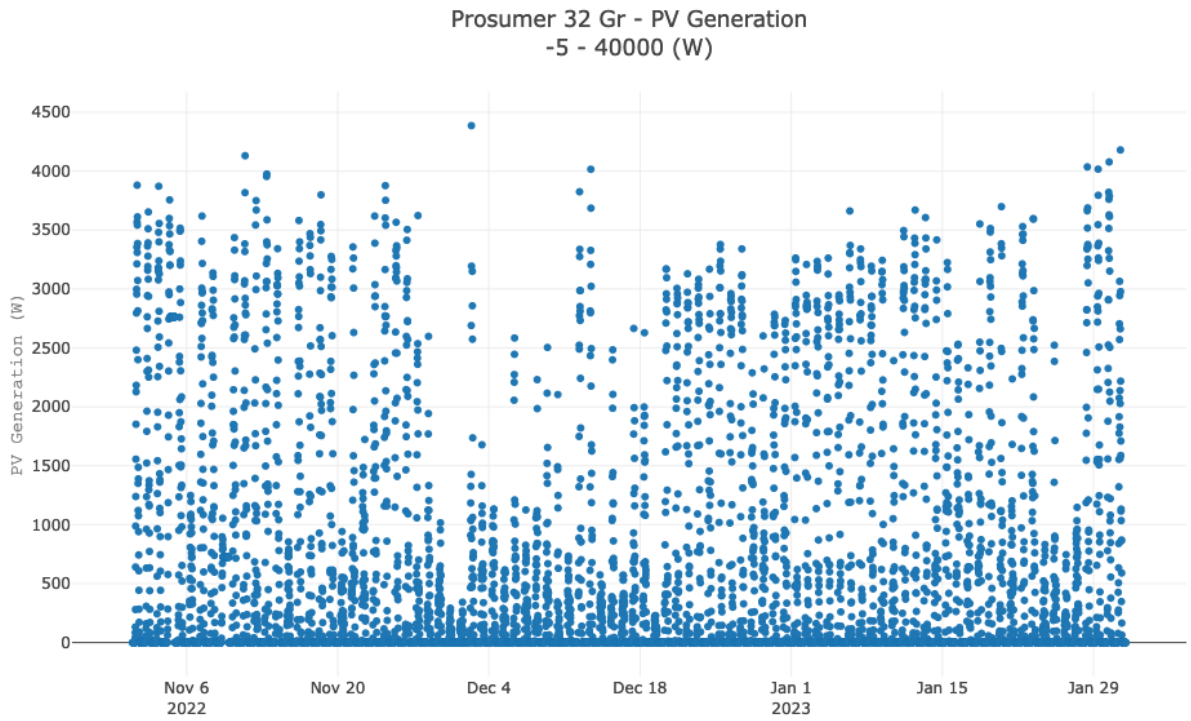

Figure 17: Μετρήσεις παραγωγής Φ/Β του 32 prosumer

4 ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

Κώδικας λήψης από smart-plug

```
#!/usr/bin/env python3
import requests
import time, datetime
import json
import math
import hamapi
import os
import sys
from datetime import datetime as dt, timedelta

API_KEY = 'api key'
DEVICES = {
    'Id1': '14:210'
}
READINGS = {
    'P': 'Power (W)',
    'V': 'Voltage (V)',
    'I': 'Current (mA)',
    'Q': 'QPower (VAR)',
    'E': 'Energy (kWh)'
```

```
}
s = "20/11/2022"
#----------------------------------#
s=f'{s} 00:00:00'
start=int(time.mktime(datetime.datetime.strptime(s,"%d/%m/%Y
%H:%M:%S").timetuple()))
end=start+86400
step=900
startdate=start
dates=[]
while True:
    if startdate > end: break
    dates.append(startdate)
    startdate=startdate+step
if name == "main":
    ham = hamapi.hamapi(api_key=API_KEY, server='hamsystems.eu')
    for name, serialno in DEVICES.items():
        header = ["Timestamp"]
        for g, dname in READINGS.items():
            header.append(dname)
        out = [';'.join(header)]
        data_pv = ham.get_datalog_data(serialno, start, end, step,
no_data_value=math.nan)
        length = len(data_pv['timestamp'])
        for i in range(length):
            timestamp = dates[i]
            #print(str(dt.fromtimestamp(timestamp).strftime('%d/%m/%y %H:%M')))
            entry = [str(dt.fromtimestamp(timestamp).strftime('%d/%m/%y %H:%M'))]
            for g in READINGS:
                entry.append(str(data_pv[g][i]))
            out.append(';'.join(entry))
        out_filename =
name+"."+str(dt.fromtimestamp(start).strftime('%d%m%y'))+'-
'+str(dt.fromtimestamp(end).strftime('%d%m%y'))+".csv"
        print(out_filename)
        with open(out_filename, 'w') as fp:
            fp.write("\n".join(out))
```

Κώδικας λήψης από Raspberry Pi

```
import csv
import time
import board
import adafruit_dht
import psutil
import os.path

for proc in psutil.process_iter():
    if proc.name() == 'libgpiod_pulsein' or proc.name() == 'libgiod_pulsei':
```

```
        proc.kill()

dhtDevice = adafruit_dht.DHT11(board.D4)

fexists = os.path.isfile('sensor_readings18.csv')
if not fexists:
    with open('sensor_readings18.csv', 'w', encoding='UTF8', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(['Date', 'Time', 'Temperature (C)',
                         'Temperature (F)', 'Humidity %'])
        f.close()

running = True
while running:
    try:
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        if humidity is not None and temperature_c is not None:
            print('Temperature = ' + str(temperature_c)+' C'+','+'Temperature
Fahrenheit =' +
            str(temperature_f)+' F'+','+'Humidity = ' + str(humidity)+'%')
            row = [time.strftime('%d/%m/%Y'), time.strftime('%H:%M:%S'),
            str(temperature_c), str(temperature_f), str(humidity)]

            with open(r'sensor_readings18.csv', 'a', newline='') as f:
                writer = csv.writer(f)
                writer.writerow(row)
            time.sleep(1)
        else:
            print('Failed to get reading. Try again')
            time.sleep(1)
    except RuntimeError as error:
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error
    except KeyboardInterrupt:
        print('Program stopped')
        running = False
    time.sleep(60.0)
```

Κώδικας λήψης από Gridvis

```
import pandas as pd
from datetime import datetime, date, timedelta
import xmltodict
import requests
import sys, os

def remove_first_end_spaces(string):
    return "".join(string.rstrip().lstrip())

def format_my_nanos(nanos):
    dt = datetime.fromtimestamp(nanos / 1e9)
    return '{}'.format(dt.strftime('%d/%m/%y %H:%M'), nanos % 1e0)

start = date.today() - timedelta(days=30)
end = date.today()  - timedelta(days=1)
start = start.strftime("EUROPEAN_%d.%m.%Y")
end = end.strftime("EUROPEAN_%d.%m.%Y")

print("start =", start)
print("end =", end)

metrics={
    "metric": {
        0: "Inverter I output-Voltage L1 ",
        1: "Inverter I output-Voltage L2",
        2: "Inverter I output-Voltage L3",
    },
    "link": {
        0:
"http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/1/hist/values/U_Eff
ective/L1/900/",
        1:
"http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/1/hist/values/U_Eff
ective/L2/900/",
        2:
"http://10.0.2.10:8180/rest/1/projects/UoWM_Residents/devices/1/hist/values/U_Eff
ective/L3/900/",
    },
}
dteliko={}
dtimestmap={}
dtimestmap['timestamp']={}

#for i in range(0,len(metrics['metric'])):
for i in metrics['metric']:

    metric=remove_first_end_spaces(metrics['metric'][i])
    link=remove_first_end_spaces(metrics['link'][i])
```

```
    if link.endswith('/'):
        link=link+'?start='+start+'&end='+end
    else:
        link=link+'&start='+start+'&end='+end

    try:
        response = requests.get(link)
        data = xmltodict.parse(response.content)

        #print('Success metric:',metric,' ',link)

        dteliko[metric]={}
        for ii in range(0,len(data['valuelist']['values'])):
            tt=format_my_nanos(int(data['valuelist']['values'][ii]['endTime']))
            if i == 0:
                dtimestmap['timestamp'][ii]=tt
            if data['valuelist']['values'][ii]['avg']!='' and
data['valuelist']['values'][ii]['avg']!='NaN':

dteliko[metric][ii]=round(float(data['valuelist']['values'][ii]['avg']),3)
            else:
                dteliko[metric][ii]=data['valuelist']['values'][ii]['avg']

    except:
        print('Error in metric:',metric,' ',link)


telikodataframe={}

telikodataframe['timestamp']=dtimestmap['timestamp']
for i in dteliko:
    telikodataframe[i]=dteliko[i]


df = pd.DataFrame(telikodataframe)
df.to_csv('gridvis_esties15m.csv', index=False, header=True)
```

Κώδικας λήψης από Inverter API

```
#!/usr/bin/env python3
import requests
import time, datetime
import json,urllib.request
import math
import sqlite3
import pandas as pd
from pprint import pprint
import os
from urllib.request import urlopen
from openpyxl import Workbook
```

```
from openpyxl import load_workbook

abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)
filename = "measurementsfronius.xlsx"
try:
    wb = load_workbook(filename)
    ws = wb.worksheets[0]  # select first worksheet
except FileNotFoundError:
    headers_row =
['Timestamp','PAC','TOTAL_ENERGY','Current_AC_Phase_1','Current_AC_Phase_2','Curr
ent_AC_Phase_3','Current_AC_Sum','EnergyReactive_VArAC_Sum_Consumed','EnergyReact
ive_VArAC_Sum_Produced','EnergyReal_WAC_Minus_Absolute','EnergyReal_WAC_Plus_Abso
lute','EnergyReal_WAC_Sum_Consumed','EnergyReal_WAC_Sum_Produced','Frequency_Phas
e_Average','Meter_Location_Current','PowerApparent_S_Phase_1','PowerApparent_S_Ph
ase_2','PowerApparent_S_Phase_3','PowerApparent_S_Sum','PowerFactor_Phase_1','Pow
erFactor_Phase_2','PowerFactor_Phase_3','PowerFactor_Sum','PowerReactive_Q_Phase_
1','PowerReactive_Q_Phase_2','PowerReactive_Q_Phase_3','PowerReactive_Q_Sum','Pow
erReal_P_Phase_1','PowerReal_P_Phase_2','PowerReal_P_Phase_3','PowerReal_P_Sum','
TimeStamp','Voltage_AC_PhaseToPhase_12','Voltage_AC_PhaseToPhase_23','Voltage_AC_
PhaseToPhase_31','Voltage_AC_Phase_1','Voltage_AC_Phase_2','Voltage_AC_Phase_3','
StateOfCharge_Relative','Temperature_Cell','Voltage_DC','E_Total','P','SOC','Back
upMode','BatteryStandby','P_Akku','P_Grid','P_Load','P_PV','rel_Autonomy','rel_Se
lfConsumption']
    wb = Workbook()
    ws = wb.active
    ws.append(headers_row)

while True:
    try:
        data1 =
pd.read_json('http://10.0.22.53/solar_api/v1/GetInverterRealtimeData.cgi') #22.53
esties
        data2 =
pd.read_json('http://10.0.22.53/solar_api/v1/GetMeterRealtimeData.cgi')
#10.0.10.41 koilada
        data3 =
pd.read_json('http://10.0.22.53/solar_api/v1/GetStorageRealtimeData.cgi')
        data4 =
pd.read_json('http://10.0.22.53/solar_api/v1/GetPowerFlowRealtimeData.fcgi')
        #1
        Timestamp=data1['Head']['Timestamp']
        PAC=data1['Body']['Data']['PAC']['Values']['1']
        TOTAL_ENERGY=data1['Body']['Data']['TOTAL_ENERGY']['Values']['1']
        #2
        Current_AC_Phase_1  =   data2   ['Body']    ['Data']    ['0']
['Current_AC_Phase_1']
        Current_AC_Phase_2  =   data2   ['Body']    ['Data']    ['0']
['Current_AC_Phase_2']
```

```
        Current_AC_Phase_3  =   data2   ['Body']    ['Data']    ['0']
['Current_AC_Phase_3']
        Current_AC_Sum  =   data2   ['Body']    ['Data']    ['0']
['Current_AC_Sum']
        EnergyReactive_VArAC_Sum_Consumed   =   data2   ['Body']    ['Data']
['0']   ['EnergyReactive_VArAC_Sum_Consumed']
        EnergyReactive_VArAC_Sum_Produced   =   data2   ['Body']    ['Data']
['0']   ['EnergyReactive_VArAC_Sum_Produced']
        EnergyReal_WAC_Minus_Absolute   =   data2   ['Body']    ['Data']    ['0']
['EnergyReal_WAC_Minus_Absolute']
        EnergyReal_WAC_Plus_Absolute    =   data2   ['Body']    ['Data']    ['0']
['EnergyReal_WAC_Plus_Absolute']
        EnergyReal_WAC_Sum_Consumed =   data2   ['Body']    ['Data']    ['0']
['EnergyReal_WAC_Sum_Consumed']
        EnergyReal_WAC_Sum_Produced =   data2   ['Body']    ['Data']    ['0']
['EnergyReal_WAC_Sum_Produced']
        Frequency_Phase_Average =   data2   ['Body']    ['Data']    ['0']
['Frequency_Phase_Average']
        Meter_Location_Current  =   data2   ['Body']    ['Data']    ['0']
['Meter_Location_Current']
        PowerApparent_S_Phase_1 =   data2   ['Body']    ['Data']    ['0']
['PowerApparent_S_Phase_1']
        PowerApparent_S_Phase_2 =   data2   ['Body']    ['Data']    ['0']
['PowerApparent_S_Phase_2']
        PowerApparent_S_Phase_3 =   data2   ['Body']    ['Data']    ['0']
['PowerApparent_S_Phase_3']
        #PowerApparent_S_Sum    =   data2   ['Body']    ['Data']    ['0']
['PowerApparent_S_Sum']
        PowerFactor_Phase_1 =   data2   ['Body']    ['Data']    ['0']
['PowerFactor_Phase_1']
        PowerFactor_Phase_2 =   data2   ['Body']    ['Data']    ['0']
['PowerFactor_Phase_2']
        PowerFactor_Phase_3 =   data2   ['Body']    ['Data']    ['0']
['PowerFactor_Phase_3']
        PowerFactor_Sum =   data2   ['Body']    ['Data']    ['0']
['PowerFactor_Sum']
        PowerReactive_Q_Phase_1 =   data2   ['Body']    ['Data']    ['0']
['PowerReactive_Q_Phase_1']
        PowerReactive_Q_Phase_2 =   data2   ['Body']    ['Data']    ['0']
['PowerReactive_Q_Phase_2']
        PowerReactive_Q_Phase_3 =   data2   ['Body']    ['Data']    ['0']
['PowerReactive_Q_Phase_3']
        #PowerReactive_Q_Sum    =   data2   ['Body']    ['Data']    ['0']
['PowerReactive_Q_Sum']
        PowerReal_P_Phase_1 =   data2   ['Body']    ['Data']    ['0']
['PowerReal_P_Phase_1']
        PowerReal_P_Phase_2 =   data2   ['Body']    ['Data']    ['0']
['PowerReal_P_Phase_2']
        PowerReal_P_Phase_3 =   data2   ['Body']    ['Data']    ['0']
['PowerReal_P_Phase_3']
```

```
        #PowerReal_P_Sum    =   data2   ['Body']    ['Data']    ['0']
['PowerReal_P_Sum']
        TimeStamp   =   data2   ['Body']    ['Data']    ['0']   ['TimeStamp']
        Voltage_AC_PhaseToPhase_12  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_PhaseToPhase_12']
        Voltage_AC_PhaseToPhase_23  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_PhaseToPhase_23']
        Voltage_AC_PhaseToPhase_31  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_PhaseToPhase_31']
        Voltage_AC_Phase_1  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_Phase_1']
        Voltage_AC_Phase_2  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_Phase_2']
        Voltage_AC_Phase_3  =   data2   ['Body']    ['Data']    ['0']
['Voltage_AC_Phase_3']
        #3
        #StateOfCharge_Relative=data3['Body']   ['Data']    ['0']
['Controller']['StateOfCharge_Relative']
        #Temperature_Cell=data3['Body'] ['Data']    ['0']
['Controller']['Temperature_Cell']
        #Voltage_DC=data3['Body']   ['Data']    ['0']
['Controller']['Voltage_DC']
        #CURRENT DC Ic

        #4
        E_Total=data4['Body']   ['Data']    ['Inverters']   ['1']['E_Total']
        P=data4['Body'] ['Data']    ['Inverters']   ['1']['P']
        SOC=data4['Body']   ['Data']    ['Inverters']   ['1']['SOC']
        BackupMode=data4['Body']    ['Data']    ['Site']    ['BackupMode']
        BatteryStandby=data4['Body']    ['Data']    ['Site']
['BatteryStandby']
        P_Akku=data4['Body']    ['Data']    ['Site']    ['P_Akku']
        P_Grid=data4['Body']    ['Data']    ['Site']    ['P_Grid']
        P_Load=data4['Body']    ['Data']    ['Site']    ['P_Load']
        #P_PV=data4['Body'] ['Data']    ['Site']    ['P_PV']
        rel_Autonomy=data4['Body']  ['Data']    ['Site']    ['rel_Autonomy']
        rel_SelfConsumption=data4['Body']   ['Data']    ['Site']
['rel_SelfConsumption']


        new_row =
[Timestamp,PAC,TOTAL_ENERGY,Current_AC_Phase_1,Current_AC_Phase_2,Current_AC_Phas
e_3,Current_AC_Sum,EnergyReactive_VArAC_Sum_Consumed,EnergyReactive_VArAC_Sum_Pro
duced,EnergyReal_WAC_Minus_Absolute,EnergyReal_WAC_Plus_Absolute,EnergyReal_WAC_S
um_Consumed,EnergyReal_WAC_Sum_Produced,Frequency_Phase_Average,Meter_Location_Cu
rrent,PowerApparent_S_Phase_1,PowerApparent_S_Phase_2,PowerApparent_S_Phase_3,Pow
erApparent_S_Sum,PowerFactor_Phase_1,PowerFactor_Phase_2,PowerFactor_Phase_3,Powe
rFactor_Sum,PowerReactive_Q_Phase_1,PowerReactive_Q_Phase_2,PowerReactive_Q_Phase
_3,PowerReactive_Q_Sum,PowerReal_P_Phase_1,PowerReal_P_Phase_2,PowerReal_P_Phase_
3,PowerReal_P_Sum,TimeStamp,Voltage_AC_PhaseToPhase_12,Voltage_AC_PhaseToPhase_23
,Voltage_AC_PhaseToPhase_31,Voltage_AC_Phase_1,Voltage_AC_Phase_2,Voltage_AC_Phas
```

```
e_3,StateOfCharge_Relative,Temperature_Cell,Voltage_DC,E_Total,P,SOC,BackupMode,B
atteryStandby,P_Akku,P_Grid,P_Load,P_PV,rel_Autonomy,rel_SelfConsumption]

        ws.append(new_row)
        wb.save(filename)
        time.sleep(5)
    except:
        print("den sindeome ston inverter")
        time.sleep(60)
```

ΑΝΑΦΟΡΕΣ

https://hamsystems.tawk.help/article/api

https://wiki.janitza.de/display/GRIDVIS70EN/REST-API

https://ecat.legrand.gr/Images/notices/LE10990AA-02.pdf