ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Σύμφωνα με το παλιό πρόγραμμα σπουδών

Εγκατάσταση συστήματος καταγραφής ηλεκτρικών δεδομένων σε μονοφασικό πίνακα και ανάπτυξη εφαρμογής Android για απομακρυσμένο έλεγχο.

Επικοινωνία μέσω Modbus.

Ελευθερία Δ.Μ. Καβουσανάκη

Αριθμός Μητρώου: 5903

Επιβλέπων καθηγητής: Καραπιδάκης Εμμανουήλ

Ηράκλειο, Σεπτέμβριος 2020

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Τμήμα Ηλεκτρολογίας

Εγκατάσταση συστήματος καταγραφής ηλεκτρικών δεδομένων σε μονοφασικό πίνακα και ανάπτυξη εφαρμογής Android για απομακρυσμένο έλεγχο.

Επικοινωνία μέσω Modbus.

Ελευθερία Δ.Μ. Καβουσανάκη

Αριθμός Μητρώου: 5903

Επιβλέπων καθηγητής: Καραπιδάκης Εμμανουήλ

Ηράκλειο, Σεπτέμβριος 2020

Πρόλογος

Η πτυχιακή αυτή έχει ως σκοπό την εγκατάσταση σε μονοφασικό πίνακα, διάταξης καταγραφής ηλεκτρικών μεγεθών (π.χ. τάση (φάση - ουδέτερος), ρεύμα γραμμής, ενεργός ισχύς, άεργος ισχύς κ.λ.π) και ανάπτυξης εφαρμογής Android για απομακρυσμένο έλεγχο. Το σύστημα αυτό βρίσκεται, ήδη, σε λειτουργία σε οικία ενώ καταγράφει δεδομένα από τις 23.08.2020. Η επιλογή της διάταξης καταγραφής και μέτρησης ηλεκτρικών δεδομένων καθώς και όλες οι υπόλοιπες διατάξεις – εφαρμογές, θα περιγραφούν αναλυτικά στις επόμενες παραγράφους. Το σύστημα χρησιμοποιεί το πρωτόκολλο Modbus προκειμένου τα δεδομένα να σταλούν από τη μετρητική διάταξη στην εφαρμογή Android, μέσω ασύρματου δικτύου. Το σύστημα που αναπτύχθηκε είναι ευέλικτο και δεν απαιτεί παρά πρόσβαση σε ένα ασύρματο δίκτυο προκειμένου να λειτουργήσει. Επιπλέον, η εφαρμογή Android υποστηρίζει παραπάνω του ενός, χρήστες ούτως ώστε να εξασφαλίζει την μέγιστη δυνατή φορητότητα.

Ευχαριστίες

Από εδώ θα ήθελα να ευχαριστήσω ειλικρινά τον επιβλέποντα καθηγητή κ. Καραπιδάκη Εμμανουήλ για το εξαιρετικό θέμα καθώς και την υπομονή του καθ' όλη τη διάρκεια της εργασίας.

Ελευθερία Δ.Μ. Καβουσανάκη, Αύγουστος 2020

Περιεχόμενα

Εισαγωγή

Η συγκεκριμένη εργασία στόχο έχει τη μέτρηση τιμών κατανάλωσης ενέργειας, τάσης φορτίου, ρεύματος γραμμής, συχνότητας, συντελεστή ισχύος κ.α. σε οικιακό περιβάλλον και την καταγραφή κι απεικόνισή τους σε περιβάλλοντα web και android. Οι τιμές αυτές όχι μόνο δίνουν πολύτιμες πληροφορίες όσον αφορά στην ενεργειακή κατανάλωση και άρα στην αναμενόμενη χρέωση από τον πάροχο ηλεκτρικής ενέργειας, αλλά, επιπλέον, δείχνουν και την ομαλότητα λειτουργίας των ηλεκτρικών συσκευών – συστημάτων των οποίων οι καταναλώσεις επιτηρούνται. Έτσι, π.χ., είναι δυνατός ο εντοπισμός ενός πιθανού εσωτερικού σφάλματος της εγκατάστασης ή μιας διακοπής στην ηλεκτροδότηση ή, ίσως, και κάποια ανωμαλία του δικτύου. Σε κάθε περίπτωση, η καταγραφή των στοιχείων ενέργειας είναι σημαντική γιατί όχι μόνο συνεισφέρει σε καλύτερη εξοικονόμηση ή στην θέσπιση και προγραμματισμό ενεργειών προς την κατεύθυνση της ελαχιστοποίησης της ενεργειακής κατανάλωσης κι, άρα, χρέωσης αλλά και γιατί μπορεί να αποτελέσει εργαλείο στην επίλυση πιθανών προβλημάτων στην ηλεκτρική εγκατάσταση.

Προκειμένου να αντιμετωπίσουμε το θέμα της παρούσας εργασίας έπρεπε να απαντήσουμε σε ορισμένα ερωτήματα μεταξύ άλλων:

1. Πώς θα μετράμε τα δεδομένα ενέργειας;
2. Πώς θα μεταφέρουμε τα δεδομένα ενέργειας στο web;
3. Η μετρητική διάταξη θα ενσωματωθεί στον ηλεκτρικό πίνακα της εγκατάστασης ή θα είναι ένα ανεξάρτητο στοιχείο με ελεύθερη θέση τοποθέτησης;
4. Τι άλλα στοιχεία θα πρέπει να είναι διαθέσιμα ώστε να εξασφαλίζεται αδιάλειπτη επικοινωνία ανάμεσα στην συσκευή μέτρησης και στο web;
5. Τα δεδομένα θα είναι προσβάσιμα στον οποιοδήποτε;
6. Η συσκευή android θα μπορεί να κρατά ιστορικό των μετρήσεων ή απλά θα δείχνει την τρέχουσα μέτρηση;

Απαντήσεις σε όλα τα παραπάνω ερωτήματα θα δοθούν στις παραγράφους που ακολουθούν. Στο τέλος θα γίνει και ένας οικονομικός απολογισμός, με όλα τα στοιχεία που αγοράστηκαν και δοκιμάστηκαν προκειμένου να φτάσουμε στο αποτέλεσμα που παρουσιάζεται εδώ.

Σκεπτικό και Δομικά Στοιχεία Συστήματος

Συνολικά, για την ανάπτυξη της τελικής εφαρμογής χρησιμοποιήθηκαν, τελικά, τα κάτωθι:

Κωδικός Υλικού	Περιγραφή	Τρόπος Σύνδεσης	Τιμή
ECR140D	Μονοφασικός Μετρητής Ενέργειας Hager	Τοποθέτηση σε ράγα του ηλεκτρικού πίνακα	89,27 €
SMC120R	Τερματική αντίσταση modbus 120 Ohm της Hager	Στα άκρα της σύνδεσης modbus	6,36 €
Arduino MKR WIFI 1010	Πλακέτα Arduino MKR με δυνατότητα σύνδεσης WiFi	Σύνδεση στην παροχή ηλεκτρικής ενέργειας	34,60 €
Arduino MKR 485 Shield	Πρόσθετη πλακέτα Arduino MKR 485 για επικοινωνία μέσω Modbus	Σύνδεση στην πλακέτα Arduino MKR WIFI 1010	34,60 €
Φορτιστής Huawei με καλώδιο USB-C	Καλώδιο παροχής ηλεκτρικής ενέργειας για την πλακέτα Arduino MKR WIFI 1010	Σύνδεση στην πλακέτα Arduino MKR WIFI 1010	12,90 €
	ΣΥΝΟΛΟ		**177.73 €**

Η βασική ιδέα της εργασίας είναι η μέτρηση στοιχείων ενέργειας και η αποθήκευσή τους σε μια βάση δεδομένων με απώτερο στόχο την μελέτη και πιθανή διάγνωση προβλημάτων. Τα δομικά στοιχεία της εργασίας αυτής θα τα δούμε αναλυτικά παρακάτω.

Μετρητής Ενέργειας

Η ανεξάρτητη μέτρηση της ηλεκτρικής ενέργειας καθώς και άλλων συναφών μεγεθών, είναι από τις πιο σημαντικές, σχεδόν αναγκαίες, λειτουργίες σε οποιαδήποτε εγκατάσταση. Μια όχι και τόσο ενδελεχής ματιά στις διαθέσιμες προτάσεις μετρητών ενέργειας είναι αρκετή να καταδείξει όχι μόνο την πληθώρα επιλογών για κάθε τύπο εγκατάστασης ή κάθε τύπο συστήματος, αλλά και την ποικιλία διαθέσιμων λειτουργιών. Στην περίπτωσή μας, σε οικιακό περιβάλλον με βασικές απαιτήσεις σε ακρίβεια και συνέπεια η επιλογή της διάταξης **ECR140D** της Hager φάνηκε ικανοποιητική. Η διάταξη αυτή, όχι μόνο μπορεί να συνδεθεί απευθείας στον ηλεκτρικό πίνακα μετά την

κεντρική ασφάλεια, αλλά έχει το πλεονέκτημα ότι τις μετρήσεις που παίρνει μπορεί να τις στείλει, μέσω του πρωτοκόλλου Modbus το οποίο θα δούμε παρακάτω, σε οποιαδήποτε συσκευή τις ζητήσει. Η ακρίβεια της συσκευής αυτής δεν είναι μεγάλη ενώ στον παρακάτω πίνακα φαίνεται η διακριτική ικανότητα της συσκευής για κάθε μέτρηση του αντίστοιχου μεγέθους:

Description	Unit	Resolution	Lengh (word)	Data Type	Function	1P_40
V(L,+1-N)	V	0,01	1	U16	H'03'	9200…27600
F	Hz	0,01	1	U16	H'03'	4500…6500
I(L1)	mA	1	2	U32	H'03'	0…40000
P(ΣL)	±KW	0,01	2	S32	H'03'	-1104…+1104
Q(ΣL)	±kvar	0,01	2	S32	H'03'	-1104…+1104
S(ΣL)	KvA	0,01	2	U32	H'03'	0…1104
PF(ΣL) IEC (§)		0,001	1	S16	H'03'	-1000…+1000
PF(ΣL) IEEE (§)		0,001	1	S16	H'03'	-1000…+1000
P(L1)	±KW	0,01	2	S32	H'03'	-1104…+1104
Q(L1)	±kvar	0,01	2	S32	H'03'	-1104…+1104
S(L1)	KvA	0,01	2	U32	H'03'	0…1104
PF(L1) IEC (§)		0,001	1	S16	H'03'	-1000…+1000
PF(L1) IEEE (§)		0,001	1	S16	H'03'	-1000…+1000
Ea+ (ΣT)	kWh	1	2	U32	H'03'	0…99,999
Er+ (ΣT)	kvarh	1	2	U32	H'03'	0…99,999
Ea- (ΣT)	kWh	1	2	U32	H'03'	0…99,999
Er- (ΣT)	kvarh	1	2	U32	H'03'	0…99,999

Παρατηρούμε ότι οι μετρήσεις της τάσης, της συχνότητας, του ρεύματος γραμμής και του συντελεστή ισχύος έχουν αρκετή ακρίβεια. Από την άλλη μεριά, οι μετρήσεις της ισχύος (ενεργού – άεργου – φαινόμενης) καθώς και οι μετρήσεις της ενέργειας (εισερχόμενης – εξερχόμενης, ενεργού – άεργου) έχουν διακριτική ικανότητα 10 W – VAR – VA και 1 kWh – kVARh αντίστοιχα. Αυτό σημαίνει ότι οι μετρήσεις της ισχύος θα είναι πολλαπλάσια του 10, ενώ η ενέργεια θα είναι ακέραιος αριθμός. Τα στοιχεία που φαίνονται στον πίνακα και αφορούν στο μήκος της λέξης (Word Length), στον τύπο της μεταβλητής (S16/32, U16/32) καθώς και στην συνάρτηση (H'03') αναφέρονται στο πρωτόκολλο Modbus και θα αναλυθούν παρακάτω. Η τελευταία στήλη του πίνακα δείχνει το εύρος τιμών κάθε μεταβλητής.

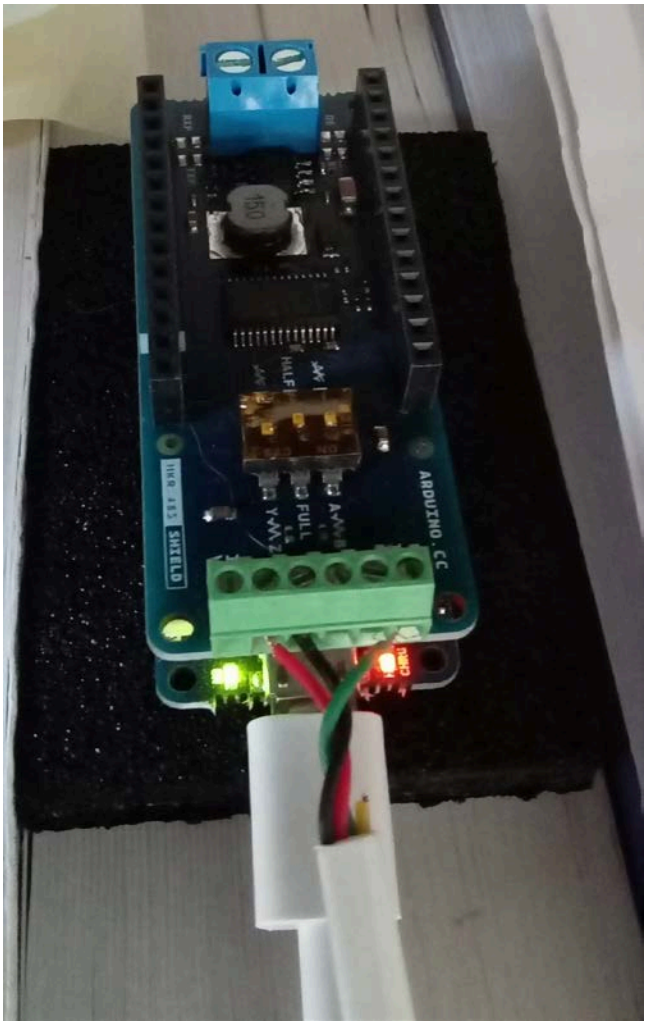
Η διάταξη συνδέθηκε στον ηλεκτρικό πίνακα μετά την ασφάλεια μιας γραμμής φωτισμού. Στην παρούσα γραμμή υπάρχουν και ρευματοδότες στους οποίους συνδέονται, κατά καιρούς, διάφορες συσκευές. Οι

μετρήσεις που έγιναν αρχικά αφορούσαν στις καταναλώσεις ενέργειας όταν σε έναν ρευματοδότη της γραμμής αυτής, συνδέθηκε ένας ανεμιστήρας.

Η επιλογή της συσκευής **ECR140D** έγινε γιατί όχι μόνο είναι ικανή να καταμετρά βασικά στοιχεία ενέργειας, αλλά και γιατί έχει τη δυνατότητα, μέσω κατάλληλης σύνδεσης, να στείλει τα δεδομένα αυτά σε όποια άλλη συσκευή τα ζητήσει. Επιπλέον, το κόστος της δεν ήταν αποτρεπτικό.

Συσκευή Ανάγνωσης δεδομένων Ενέργειας

Δεδομένης της δυνατότητας ανάγνωσης των στοιχείων ενέργειας από τη μετρητική διάταξη **ECR140D**, ήταν αναγκαία η χρησιμοποίηση κάποιας συσκευής η οποία θα μπορεί να τα διαβάσει και να τα στείλει, ιδανικά, σε κάποιον διακομιστή. Μετά από ενδελεχή έρευνα των διαθέσιμων επιλογών, διαπιστώθηκε ότι μια κατασκευή βασισμένη σε Arduino θα μπορούσε αφενός να λύσει το πρόβλημα της ανάγνωσης των στοιχείων από την μετρητική διάταξη αλλά και να παράσχει την συνδεσιμότητα με κάποιο διαθέσιμο διακομιστή σε λογικό κόστος.



Μετά από ενδελεχή έρευνα αγοράς, επιλέχθηκε η διάταξη Arduino MKR WiFi 1010 πάνω στην οποία τοποθετήθηκε ένα Arduino MKR 485 Shield. Η πρώτη διάταξη επιτρέπει τη συνδεσιμότητα με το internet καθώς και την αδιάλειπτη λειτουργία, καθώς είναι αυτή που συνδέεται με την παροχή ηλεκτρικής ενέργειας, ενώ η δεύτερη επιτυγχάνει τη σύνδεση με το ECR140D μέσω του πρωτοκόλλου modbus. Προκειμένου να συνδεθεί στην παροχή ηλεκτρικής ενέργειας το Arduino MKR WiFi 1010, απαιτείται η αγορά ενός φορτιστή USB-C σε Wall Mounted Adapter. Επιπλέον, σύμφωνα με τις οδηγίες που παρέχονται από την Hager για τον ECR140D, προκειμένου να επιτευχθεί η σύνδεση μέσω modbus του ενεργειακού μετρητή με οποιαδήποτε συμβατή συσκευή, θα πρέπει, στα σημεία σύνδεσης, να τοποθετηθεί μια αντίσταση 120 Ohm. Οι τερματικές αντιστάσεις έχουν στόχο να ελαχιστοποιήσουν τον θόρυβο σε περίπτωση που το καλώδιο μεταφοράς δεδομένων είναι μεγάλου

μήκους ή χρησιμοποιείται υψηλή ταχύτητα αποστολής δεδομένων (baud rate). Στην περίπτωσή μας, το baud rate ήταν 9600 bps ενώ το καλώδιο σύνδεσης του Arduino MKR 485 Shield με το **ECR140D** δεν ξεπερνούσε το 1 μέτρο. Συνεπώς, η τερματική αντίσταση δεν ήταν απαραίτητη. Το καλώδιο που χρησιμοποιήθηκε προκειμένου να συνδεθεί το ECR140D με το Arduino MKR 485 Shield, ήταν πλακέ τηλεφωνικό τεσσάρων αγωγών.

Η σύνδεση επικοινωνίας είναι Half Duplex, με την έννοια ότι το Arduino λειτουργεί ως Client – Master ενώ ο ενεργειακός μετρητής ως Server – Slave και άρα το Arduino στέλνει ερωτήσεις σε και δέχεται αποκρίσεις από τον ενεργειακό μετρητή. Σε καμιά περίπτωση δεν ελέγχει ο ενεργειακός μετρητής την ύπαρξη ή όχι του Arduino ή αποπειράται να εκκινήσει ο ίδιος την επικοινωνία. Ο ενεργειακός μετρητής είναι ένα παθητικό στοιχείο που αποκρίνεται μόνο όταν του τίθενται οι κατάλληλες ερωτήσεις. Το Arduino MKR 485 Shield έχει τη δυνατότητα και της Full Duplex επικοινωνίας, να χρησιμοποιείται το ίδιο κανάλι επικοινωνίας για ταυτόχρονη αποστολή και λήψη στοιχείων, αλλά αυτό στην περίπτωσή μας είναι περιττό μια που, όπως, αναφέραμε, ο ενεργειακός μετρητής απαντά αφού πρώτα του τεθεί η ερώτηση. Το Arduino θα επαναλάβει τη διαδικασία ερωταποκρίσεων μόνο όταν έχει ήδη λάβει την απάντηση του μετρητή ενέργειας. Η ρύθμιση για την Half Duplex επικοινωνία στο Arduino MKR 485 Shield γίνεται θέτοντας το FULL σε κατάσταση OFF.

Συσκευή Αποστολής δεδομένων Ενέργειας

Το Arduino MKR 485 Shield έχει τοποθετηθεί πάνω στο Arduino MKR WiFi 1010 το οποίο και αποστέλλει τα στοιχεία ενέργειας που διαβάζονται από το **ECR140D.** Η σύνδεση με το internet γίνεται μέσω WiFi. Στην προκειμένη περίπτωση η σύνδεση γίνεται με το τοπικό WiFi, ενώ σε περίπτωση εγκατάστασης όπου το WiFi είναι διαφορετικού SSID, θα πρέπει να γίνεται επανεκκίνηση του Arduino MKR WiFi 1010 ώστε να διαβαστεί το νέο SSID. Επίσης, σε περίπτωση που για κάποιο λόγο η σύνδεση με το WiFi διακοπεί, θα πρέπει, πάλι να γίνει επανεκκίνηση του Arduino MKR WiFi 1010. Δυστυχώς, αυτό είναι μια δυσχέρεια των βιβλιοθηκών του Arduino, κι ενώ υπάρχει κάποιος τρόπος, προγραμματιστικά (μέσω π.χ. του watchdog timer) να ξεπεραστεί, εντούτοις δεν είναι αρκετά ικανοποιητικός για να μπορέσει το Arduino MKR WiFi 1010 να αποτελέσει π.χ. λύση σε περιβάλλοντα όπου η αδιάλειπτη λειτουργία είναι ζωτικής σημασίας. Σε οικιακό χώρο, όμως, ανταποκρίνεται λίαν ικανοποιητικά. Επίσης, δεν υπάρχει και τρόπος ώστε η επανεκκίνηση να γίνεται από απόσταση ενώ, προς το παρόν, σε περίπτωση κάποιας καθυστέρησης λήψης ή αποστολής δεδομένων, θα πρέπει να γίνεται διακοπή της τροφοδοσίας με φυσική παρουσία στο χώρο. Με την επανεκκίνηση της τροφοδοσίας, ξεκινά να τρέχει και το εγκατεστημένο πρόγραμμα στην πλακέτα.

Ανάγκη δημιουργίας συστήματος

Το ζήτημα του ελέγχου της ενεργειακής κατανάλωσης είναι εξαιρετικής σημασίας όχι μόνο για οικονομικούς λόγους ή για λόγους διάγνωσης πιθανών προβλημάτων στην εγκατάσταση ή στην τροφοδοσία. Η ενεργειακή κατανάλωση και ο έλεγχός της είναι βασικό στοιχείο μιας περιβαλλοντικής προσέγγισης και μιας φιλικότερης προς το περιβάλλον στάσης μια που η ενέργεια όχι μόνο δεν είναι απεριόριστη αλλά έχει σημαντικό κόστος, κυρίως, στο περιβάλλον. Το παρόν σύστημα, αν και αποτελεί μια αρκετά περιορισμένη ενεργειακή λύση, εντούτοις μπορεί να δώσει σημαντικές πληροφορίες αναφορικά με την κατανάλωση, την κατανομή του φορτίου και, ασφαλώς, πηγές πιθανού σφάλματος. Το παρόν σύστημα, δίνει σημαντικές πληροφορίες στον χρήστη για την κατανάλωσή του 24/7 με πολύ οικονομικό και συστηματικό τρόπο. Η εγκατάσταση στο σπίτι είναι εξαιρετικά εύκολη και δεν απαιτεί πολύπλοκες μετατροπές στην ήδη υπάρχουσα εγκατάσταση. Αν και οι μετρήσεις πάνω στον ενεργειακό μετρητή και αυτές που διαβάζει στο σύστημα μέσω modbus δεν έχουν την ίδια ακρίβεια, π.χ. η συνολική ενέργεια έχει ακρίβεια ακεραίου αν διαβάζεται μέσω modbus ενώ πάνω στον μετρητή έχει ακρίβεια δεύτερου δεκαδικού ψηφίου, εντούτοις το συνολικό σφάλμα στην εκτίμηση της κατανάλωσης δεν ξεπερνά το 1%.

Τμήματα συστήματος – Καταμερισμός Εργασιών

Το σύστημα που αναπτύξαμε αποτελείται από 3 βασικά κομμάτια:

1. Ο ενεργειακός μετρητής
2. Η συσκευή ανάγνωσης μέσω modbus από τον ενεργειακό μετρητή και αποστολής των στοιχείων αυτών μέσω wifi
3. Οι πλατφόρμες στις οποίες μπορεί να συνδεθεί ο χρήστης προκειμένου να δει τα αποθηκευμένα στοιχεία είτε μέσω ενός web portal είτε μέσω μιας εφαρμογής android.

Ο ενεργειακός μετρητής συνδέεται απευθείας σε μια φάση στον πίνακα του σπιτιού. Στην περίπτωσή μας συνδέθηκε με μια γραμμή φωτισμού στην οποία υπάρχει και ένας ρευματοδότης.

Η συσκευή ανάγνωσης των στοιχείων του ενεργειακού μετρητή είναι ένα arduino το οποίο, όχι μόνο διαβάζει μέσω modbus, τα στοιχεία του ενεργειακού μετρητή αλλά τα στέλνει και σε μια διαδικτυακή βάση δεδομένων από την οποία μπορεί να τα αντλήσει ο χρήστης και να τα μελετήσει. Η σύνδεση με το internet γίνεται ασύρματα.

Τέλος, προκειμένου τα δεδομένα να είναι αναγνώσιμα και προσβάσιμα στον μέσο χρήστη, αναπτύχθηκαν αφενός μια διαδικτυακή πλατφόρμα προκειμένου τα δεδομένα να είναι προσβάσιμα με το σκεπτικό ότι θα μπορούσε π.χ. η διάταξή μας να εγκατασταθεί σε διαφορετικές τοποθεσίες και αφορά διαφορετικούς χρήστες ενώ, επίσης, αναπτύχθηκε μια εφαρμογή android που διαβάζει τα

σχετικά με τον συγκεκριμένο χρήστη, στοιχεία ενώ θα μπορούσε να χρησιμοποιηθεί από διαφορετικούς χρήστες για διαφορετικά στοιχεία.

Προγραμματισμός – Συμβατότητα

Η διαδικτυακή πλατφόρμα αναπτύχθηκε σε PHP. Αυτή είναι η βασική γλώσσα προγραμματισμού για διαδικτυακές εφαρμογές που λειτουργούν μέσω του φυλλομετρητή. Επίσης, είναι η βασική γλώσσα προγραμματισμού για πλατφόρμες που στηρίζονται σε βάσεις δεδομένων. Η βάση δεδομένων που χρησιμοποιούμε είναι MySQL. Η πλατφόρμα που αναπτύξαμε είναι συμβατή με κάθε φυλλομετρητή ενώ είναι προσβάσιμη και από φυλλομετρητές που λειτουργούν σε κινητά τηλέφωνα. Όμως, επειδή οι συσκευές αυτές έχουν περιορισμένες διαστάσεις, μπορεί η πρόσβαση στην πλατφόρμα αλλά και οι πληροφορίες που αυτή περιέχει να μην γίνονται εύκολα αντιληπτές. Για το λόγο αυτό, αναπτύχθηκε και μια εφαρμογή android η οποία όχι μόνο λύνει το θέμα της προσβασιμότητας και της αναγνωσιμότητας αλλά προσφέρει και τη δυνατότητα υπολογισμού βασικών στοιχείων για την κατανάλωση όπως π.χ. το αναμενόμενο κόστος σύμφωνα με την προκαθορισμένη από τον χρήστη τιμή της κιλοβατώρας, αλλά και δίνει βασικά μηνιαία στατιστικά στοιχεία όπως μέγιστη ισχύ, μέγιστο κι ελάχιστο ρεύμα, μέγιστη κι ελάχιστη συχνότητα, ώστε ο χρήστης να μπορεί να βλέπει ανά πάσα στιγμή ποιά είναι τα εύρη των βασικών μεγεθών της παροχής του και να διαγιγνώσκει πιθανά προβλήματα εάν π.χ. αυτά φανούν εξαιρετικά διευρυμένα. Η εφαρμογή του αναπτύχθηκε γράφτηκε για κινητά τηλέφωνα που “τρέχουν” το λειτουργικό android κυρίως γιατί αυτά είναι τα πιο διαδεδομένα κι αφετέρου γιατί το android ως περιβάλλον προγραμματισμού έχει πολλές ομοιότητες με περιβάλλοντα όπου αναπτύσσονται εφαρμογές Java. Επίσης, έχει μεγάλη βιβλιοθήκη με εκτεταμένες περιγραφές των λειτουργιών των συναρτήσεων που διαθέτει ενώ η κοινότητα προγραμματιστών που δίνει απαντήσεις σε πιθανά ερωτήματα είναι ισχυρή και μεγάλη.

Τέλος, η συσκευή arduino που ουσιαστικά εκτελεί την πιο δύσκολη εργασία της ανάγνωσης των στοιχείων από τον ενεργειακό μετρητή και της αποστολής των, μέσω wifi, προγραμματίστηκε μέσω του περιβάλλοντος Arduino και η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι, βασικά, η C. Το πρόγραμμα που γράφτηκε για τη συσκευή arduino χρησιμοποιεί βιβλιοθήκες της γλώσσας C ενώ διαθέτει και κάποιες λειτουργίες σε επίπεδο bit που είναι απαραίτητες για την σωστή ανάγνωση και απεικόνιση των δεδομένων μέσω modbus.

Συνεπώς, για την εργασία αυτή αναπτύχθηκαν:

1. Μια πλατφόρμα web στην οποία φαίνονται τα στοιχεία του μετρητή ενέργειας ανά χρήστη.
2. Μια εφαρμογή android στην οποία φαίνονται τα στοιχεία του μετρητή ενέργειας ανά χρήστη αλλά και συγκεντρωτικά στοιχεία κατανάλωσης – παροχής όπως και εκτιμήσεις κόστους

3. Μια εφαρμογή arduino η οποία διαβάζει τα στοιχεία από τον ενεργειακό μετρητή, τα μετατρέπει σε κατάλληλη μορφή μέσω πράξεων σε επίπεδο bit, και τα αποστέλλει μέσω wifi στην διαδικτυακή βάση που αναπτύχθηκε στο στάδιο 1.

Και οι τρεις εφαρμογές που αναπτύχθηκαν δεν αφορούν αποκλειστικά την συγκεκριμένη διάταξη – σύστημα που αναπτύξαμε αλλά έχουν τη δυνατότητα να εξυπηρετήσουν στην ίδια μορφή, θεωρητικά αμέτρητους χρήστες. Αυτό, μας δίνει την ελπίδα ότι, εφόσον το σύστημα αποδειχτεί αξιόπιστο και μη ευάλωτο σε αυξομειώσεις τάσης ή διακοπής της παροχής, να μπορεί να οδηγήσει σε κάποιο εμπορικό πακέτο.

Μελλοντική Επέκταση Κώδικα

Ο κώδικας, όπως είναι τώρα, μπορεί να υποστηρίξει ένα (1) ενεργειακό μετρητή ανά χρήστη. Εφόσον, ο χρήστης, όμως, αποφασίσει ότι θέλει να έχει ξεχωριστούς ενεργειακούς μετρητές για διαφορετικές γραμμές του σπιτιού του ή διαφορετικούς ενεργειακούς μετρητές για διαφορετικές τοποθεσίες τις οποίες θα ήθελε να παρακολουθεί, αυτή η εφαρμογή δεν τον καλύπτει. Θα πρέπει να γίνουν σημαντικές αλλαγές και στον κώδικα της πλατφόρμας αλλά και στην εφαρμογή android προκειμένου να εξυπηρετηθεί τυχαίος αριθμός από ενεργειακούς μετρητές. Η μόνη εφαρμογή που θα παραμείνει αμετάβλητη είναι η εφαρμογή arduino με την προϋπόθεση ότι οι παραπάνω ενεργειακοί μετρητές θα είναι όλοι το ίδιο μοντέλο και θα έχουν την ίδια βιβλιοθήκη καταχωρητών.

Μια μελλοντική σκέψη μας είναι να μπορούμε να εξυπηρετήσουμε παραπάνω του ενός ενεργειακούς μετρητές αλλά θα πρέπει πρώτα να απαντήσουμε σε ερωτήσεις όπως π.χ. αν τα συγκεντρωτικά στοιχεία ή τα μηνιαία στατιστικά θα πρέπει να παρουσιάζονται ανά μετρητή ή ανά εγκατάσταση (π.χ. οικία), ποια η συχνότητα ανάγνωσης των στοιχείων του μετρητή και ποιό το κριτήριο προκειμένου να ξεκινήσει η ανάγνωση (π.χ. μια αλλαγή στην τάση τροφοδοσίας ή μια αλλαγή στο ρεύμα γραμμής θα μπορούσε να αποτελέσει αιτία εκκίνησης της ανάγνωσης των στοιχείων του μετρητή;). Στην εφαρμογή όπως την αναπτύξαμε έως τώρα, η ανάγνωση των στοιχείων γίνεται ανά 10 λεπτά χωρίς κάποια πιο “έξυπνη” προσέγγιση όπως π.χ η ανίχνευση κάποιας αλλαγής στο κύκλωμα. Θα μπορούσαμε, ίσως, στο μέλλον να ενσωματώσουμε διαδικασίες όπως ο εντοπισμός τροποποιήσεων είτε στην τροφοδοσία είτε στην κατανάλωση οι οποίες και να αποτελούν την αιτία της εκκίνησης ενός νέου κύκλου ανάγνωσης στοιχείων.

Ιστοσελίδα

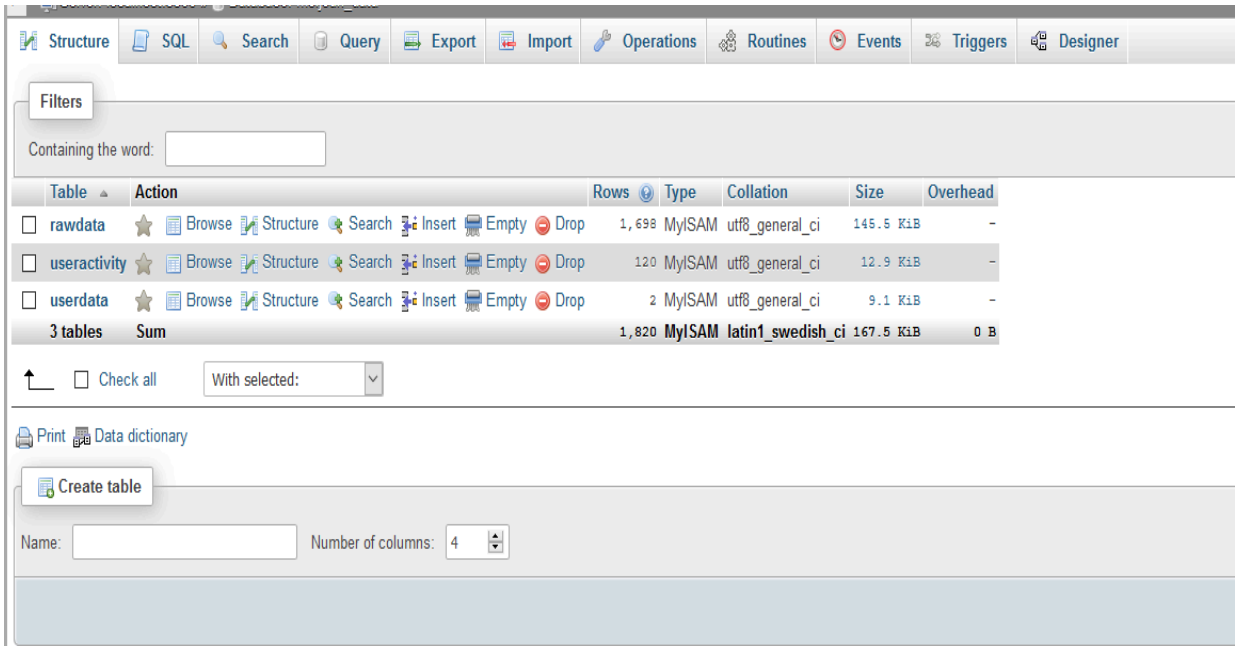
Η ιστοσελίδα σχεδιάστηκε ώστε να δείχνει την τελευταία μέτρηση του μετρητή ενέργειας, στον χρήστη. Βασίζεται σε κώδικα PHP ενώ δίνει τη δυνατότητα σε διάφορους χρήστες να έχουν δικό τους

λογαριασμό όπου θα βλέπουν ανά πάσα ώρα και στιγμή τα δεδομένα του δικού τους ενεργειακού μετρητή.

Υπόβαθρο

Η σελίδα έχει από πίσω μια βάση δεδομένων MySQL από την οποία φορτώνονται τα τελευταία στοιχεία με ημερολογιακή σειρά. Η βάση δεδομένων χρησιμοποιείται και από την εφαρμογή android που θα δούμε παρακάτω και η οποία αποτελείται από 3 βασικούς πίνακες:

1. Τον πίνακα αποθήκευσης των δεδομένων ενέργειας από τον μετρητή (rawdata)
2. Τον πίνακα αποθήκευσης των στοιχείων του χρήστη (userdata)
3. Τον πίνακα αποθήκευσης των στοιχείων πρόσβασης του εκάστοτε χρήστη – για διάγνωση πιθανών προβλημάτων ή παραβιάσεων (useractivity)



Τα στοιχεία που αποθηκεύονται στη βάση ταξινομούνται, πολύ εύκολα, ανά χρήστη. Κάθε πίνακας της βάσης έχει τις παρακάτω στήλες:

1. Rawdata

i. update_date: η ημερομηνία και η ώρα της καταχώρησης στην βάση – αυτόματη πληροφορία κατά την εγγραφή των στοιχείων ενέργειας

ii. email: το μοναδικό διακριτικό για τον εκάστοτε χρήστη που εκτελεί χρέη αναγνωριστικού (username)

iii. active_energy_in: η ενέργεια που καταναλώθηκε από τον χρήστη όπως δίνεται από τον ενεργειακό μετρητή σε kWh

iv. active_energy_out: η ενέργεια που προσφέρθηκε από τον χρήστη στο δίκτυο σε kWh. Για την περίπτωσή μας αυτό δεν είναι ένα στοιχείο που μας αφορά.
v. active_power: η ισχύς που καταναλώνεται σε W
vi. voltage: η τάση στα άκρα της γραμμής σε V
vii. current: το ρεύμα γραμμής σε A
viii. frequency: η συχνότητα του εναλλασσόμενου ρεύματος σε Hz
ix. power_factor: ο συντελεστής ισχύος (καθαρός αριθμός)

Η ισχύς που καταναλώνεται μπορεί και να υπολογιστεί από το γινόμενο:

$$P = V_{\text{rms}} \cdot I_{\text{rms}} \cdot \cos\theta$$

Θα δούμε ότι το γινόμενο αυτό και τα στοιχεία του μετρητή δίνουν συμβατά αποτελέσματα.

1. Useractivity
   i. email: το μοναδικό διακριτικό για τον εκάστοτε χρήστη που εκτελεί χρέη αναγνωριστικού (username)
   ii. session_id: το μοναδικό διακριτικό για την εκάστοτε συνεδρία που δημιουργεί ο χρήστης στον browser με την πλατφόρμα
   iii. lastlogin: η τελευταία φορά που ο χρήστης συνδέθηκε στην ιστοσελίδα ή η τελευταία φορά που αποσυνδέθηκε από αυτή
   iv. currentstatus: η κατάσταση του χρήστη (σε σύνδεση ή όχι)

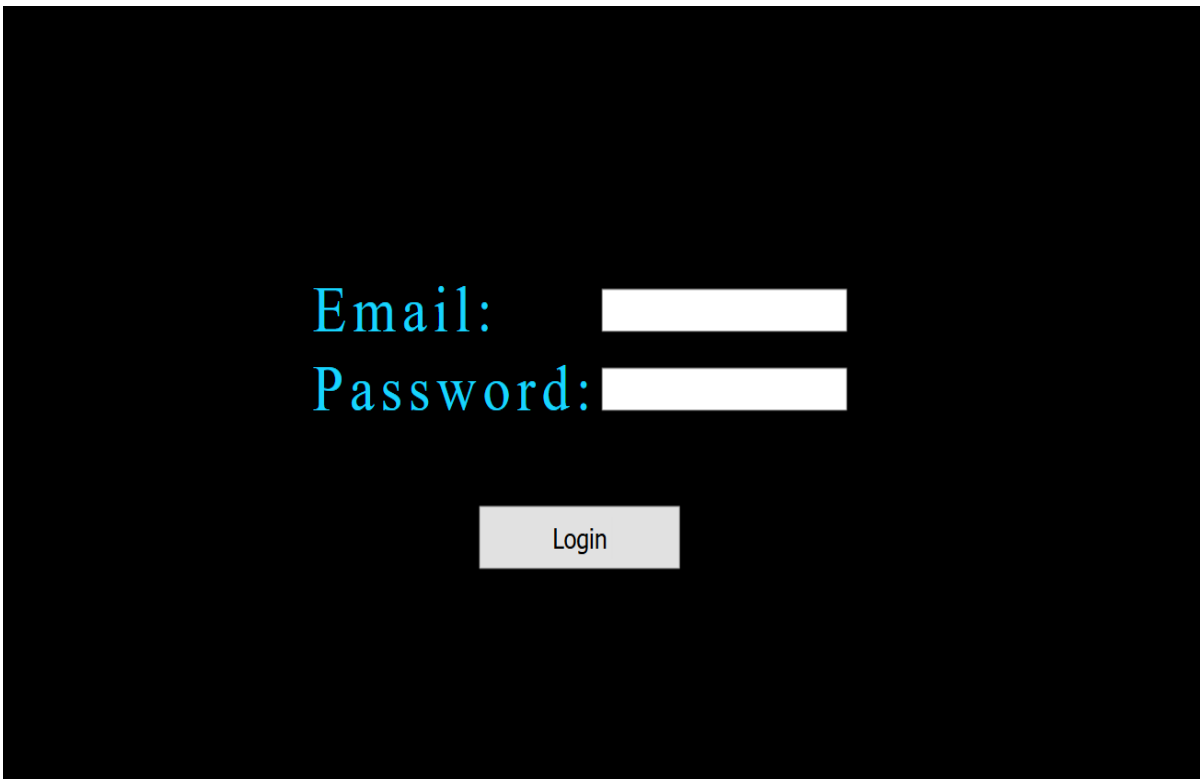
Όπως είπαμε, ο πίνακας αυτός χρησιμοποιείται μόνο για διαγνωστικούς λόγους από τον webmaster.

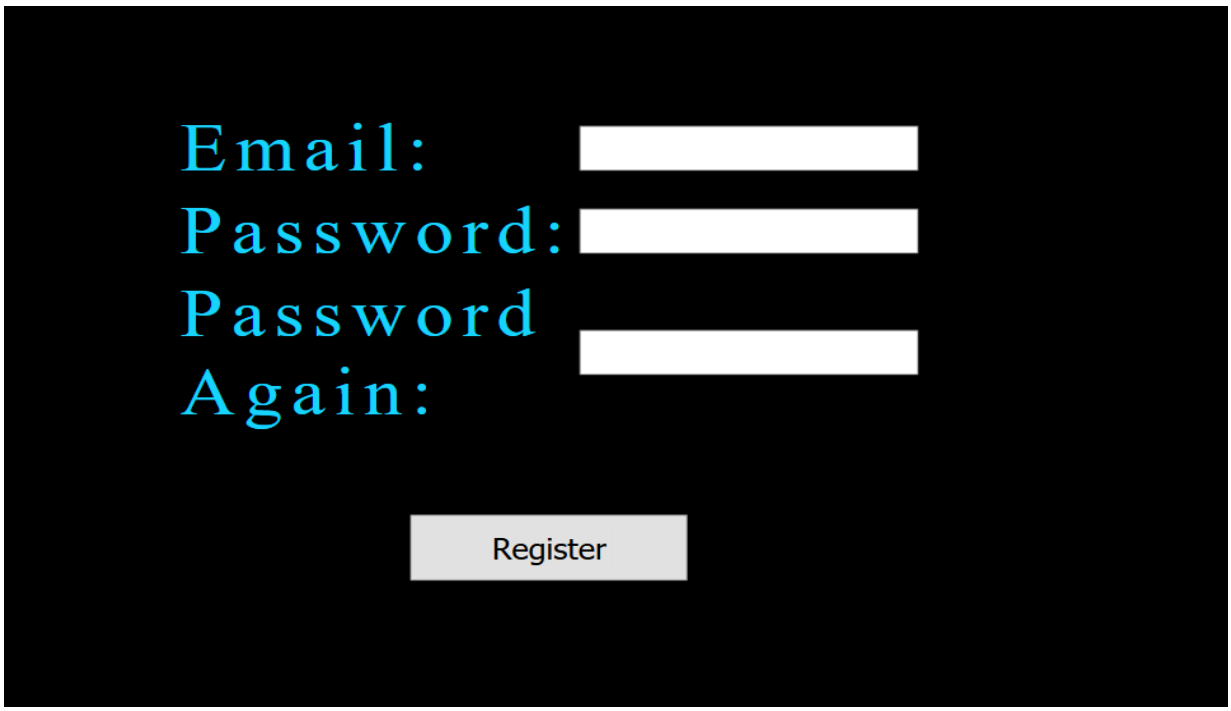
1. Userdata
   1. email: το μοναδικό διακριτικό για τον εκάστοτε χρήστη που εκτελεί χρέη αναγνωριστικού (username)
   2. password: το συνθηματικό του εκάστοτε χρήστη. Στη βάση έχει κωδικοποιηθεί με τον κώδικα md5 που παράγει 128 bit κωδικοποιημένες συμβολοσειρές.

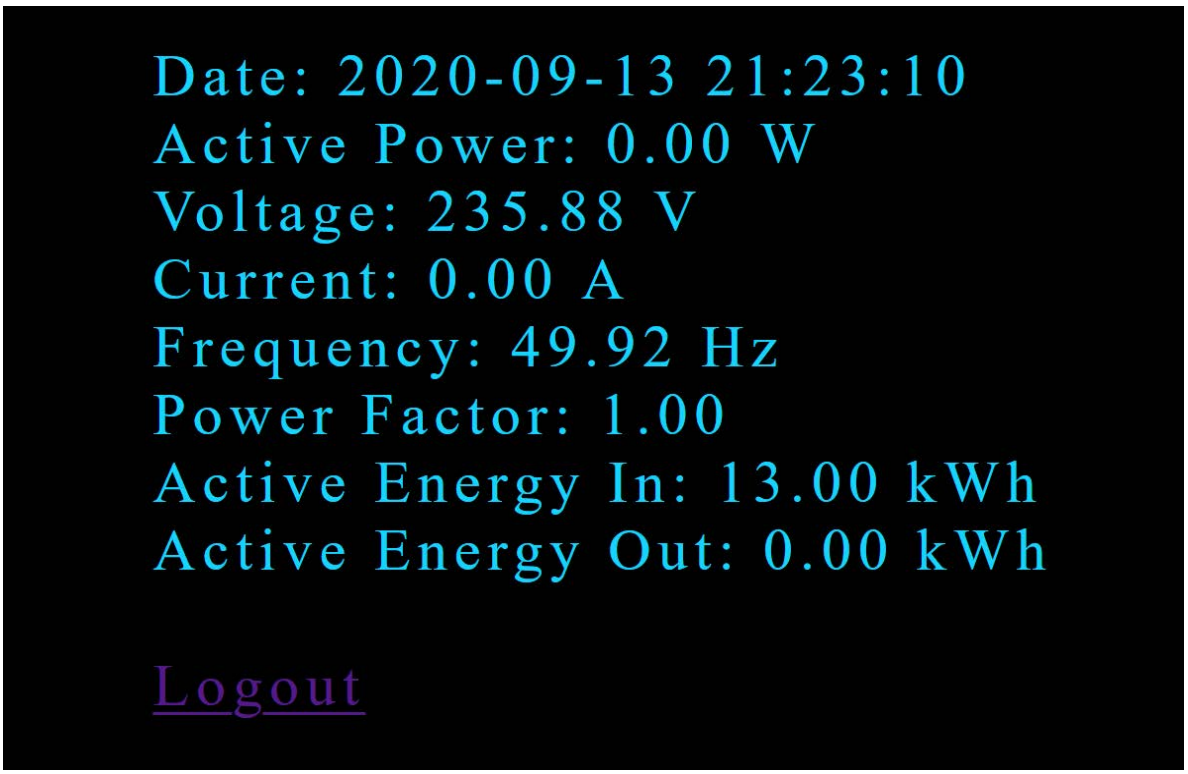
Στοιχεία Σελίδας

Η σελίδα αποτελείται από την αρχική επιβεβαίωση των στοιχείων του χρήστη ή την παροχή της δυνατότητας δημιουργίας νέου χρήστη στην πλατφόρμα ενώ δίνεται και η δυνατότητα αναθεώρησης του συνθηματικού του χρήστη σε περίπτωση απώλειας. Μετά την είσοδο του χρήστη στην πλατφόρμα, φαίνεται απευθείας η τελευταία καταχωρημένη μέτρηση από τον μετρητή ενέργειας. Η ανανέωση των στοιχείων γίνεται αυτόματα κάθε λεπτό. Ο χρήστης μπορεί να δει όλες τις εγγραφές της βάσης, ταξινομημένες ανά μήνα, μόνο μέσω της εφαρμογής android. Αυτό έγινε γιατί αφενός η διαδικτυακή

πλατφόρμα, ουσιαστικά, έχει στόχο τη δημιουργία του χρήστη κι αφετέρου γιατί η εφαρμογή android έχει πολύ μεγαλύτερη ευελιξία από μια διαδικτυακή εφαρμογή στηριγμένη σε φυλλομετρητή.







Η τελευταία εικόνα αφορά τις μετρήσεις που έστειλε το arduino στον διακομιστή (server) όπως τις διάβασε από τον ενεργειακό μετρητή. Τη δεδομένη χρονική στιγμή δεν υπάρχει κάποιο φωτιστικό στοιχείο που να τραβάει ενέργεια ή κάποια άλλη διάταξη συνδεδεμένη στον ρευματοδότη της γραμμής. Όλος ο κώδικας των σελίδων που φτιάξαμε βρίσκεται στο παράρτημα της εργασίας αυτής.

Modbus

Το Modbus είναι ένα πρωτόκολλο επικοινωνίας που αναπτύχθηκε το 1979 από την εταιρεία Modicon (τώρα Schneider Electric) και το οποίο είχε στόχο την εδραίωση αμφίδρομης επικοινωνίας ανάμεσα σε έξυπνες συσκευές με τη λογική του “master” – “slave”: η μία συσκευή να θέτει τις ερωτήσεις και η άλλη ή οι άλλες να απαντούν. Αρχικά, σε ένα κύκλωμα modbus, μπορούσε να υπάρχει μόνο μια συσκευή “master” που θα έθετε τις ερωτήσεις και διάφορες συσκευές “slave” που θα απαντούσαν. Το πρωτόκολλο modbus συνδέεται μέσω σειριακής επικοινωνίας βασισμένης στο RS-485 στάνταρντ. Το τελευταίο καθορίζει τον τύπο του καλωδίου αλλά και την μέγιστη ταχύτητα επικοινωνίας ανάμεσα στις διατάξεις, δεδομένου του μήκους της γραμμής σύνδεσης. Είναι το αντίστοιχο για το TCP/IP δίκτυο του RJ-45. Το στάνταρντ RS-485 επιτρέπει τη σύνδεση μιας συσκευής με πολλές άλλες, τη σύνδεσή τους έως και 1.2 km μακρυά ενώ ελαχιστοποιεί τον ηλεκτρομαγνητικό θόρυβο. Συνεπώς, το RS-485 είναι ο τρόπος σύνδεσης συσκευών που επικοινωνούν μέσω του πρωτοκόλλου modbus. Οι συσκευές μπορούν να επικοινωνούν αμφίδρομα και ταυτόχρονα αλλά για να επιτευχθεί μια τέτοια επικοινωνία απαιτούνται 4 καλώδια αντί για 3 στην περίπτωση της ασύγχρονης επικοινωνίας. Τα καλώδια που θα χρειαστούμε εμείς είναι 2 για την επικοινωνία (+ / -) και 1 για τη γείωση (0). Στην περίπτωση που θέλαμε σύγχρονη επικοινωνία θα έπρεπε να είχαμε και κάποια άλλη υποδοχή για τον 4ο καλώδιο. Ο ενεργειακός μετρητής, μας, όμως, δεν παρέχει τέτοια δυνατότητα και άρα η επικοινωνία μας θα είναι ασύγχρονη ή, όπως λέγεται στην ορολογία modbus: half duplex.

Επίσης, στην περίπτωση που είχαμε παραπάνω του ενός μετρητές ενέργειας και θα θέλαμε να διαβάζαμε τα περιεχόμενά τους με την ίδια διάταξη arduino θα έπρεπε να επιτελέσουμε την modbus συνδεσμολογία σε παράλληλη διάταξη ώστε κάθε (+) ακροδέκτης της modbus επικοινωνίας να είναι συνδεδεμένος με τους υπόλοιπους (+) ακροδέκτες και αντίστοιχα, κάθε (-) ακροδέκτης της modbus επικοινωνίας να είναι συνδεδεμένος με τους υπόλοιπους (-) ακροδέκτες. Σε καμιά περίπτωση δεν πρέπει να συγχέουμε την συνδεσμολογία modbus με την συνδεσμολογία ισχύος.

Προκειμένου να συνδεθεί ο ενεργειακός μετρητής και το arduino το οποίο θα διαβάσει και τα δεδομένα από τον μετρητή μέσω modbus, θα πρέπει να καθοριστούν τα παρακάτω:

1. Ταχύτητα επικοινωνίας στο κανάλι μετάδοσης πληροφορίας (Baud Rate) : Μπορεί να είναι 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps kai 38400 bps (τουλάχιστον για τη διάταξη που χρησιμοποιούμε εμείς)
2. Αριθμός δεδομένων σε bit, που θα αποστέλλονται με κάθε πακέτο (data bits) : Μπορεί να είναι 7 ή 8
3. Αριθμός bit που θα αποτελούν το τέλος κάθε “φράσης” που θα αποστέλλεται (stop bits): Μπορεί να είναι 1 ή 2

4. Ισοτιμία (parity) που ελέγχει εάν τα δεδομένα έχουν σταλεί σωστά και ολοκληρωμένα. Μπορεί να πάρει τιμές: άρτια (even), περιττή (odd), καθόλου (none).

Ο ενεργειακός μετρητής έχει ήδη από το εργοστάσιο, καθορισμένα τα παρακάτω:

1. Ταχύτητα επικοινωνίας : 19200 bps
2. Αριθμός δεδομένων πακέτου : 8 bits
3. Αριθμός bit τέλους : 1
4. Ισοτιμία : Άρτια

Προκειμένου, όμως, να συνδεθεί με τη διάταξη arduino η οποία και θα διαβάζει τα δεδομένα, θα πρέπει να γίνουν κάποιες αλλαγές στις προκαθορισμένες ρυθμίσεις παραπάνω. Έτσι καθορίστηκαν:

1. Ταχύτητα επικοινωνίας : 9600 bps
2. Αριθμός δεδομένων πακέτου : 8 bits
3. Αριθμός bit τέλους : 1
4. Ισοτιμία : Καθόλου

Επίσης, θα πρέπει να καθοριστεί και η μοναδική διεύθυνση της διάταξής μας (ενεργειακός μετρητής). Αυτή αφέθηκε στην προκαθορισμένη ρύθμιση που αντιστοιχούσε στον αριθμό 1.

Εφόσον, στο μέλλον, συνδέσουμε κι άλλη διάταξη στο δίκτυο modbus, θα πρέπει να έχουμε υπόψη μας τις παραπάνω ρυθμίσεις τις οποίες θα πρέπει να εφαρμόσουμε στη διάταξη αυτή απαραίτητα προκειμένου να μπορέσουμε να επικοινωνήσουμε μαζί της.

Εμείς θα διαβάσουμε τις παρακάτω μεταβλητές από τον ενεργειακό μετρητή:

1. V(L1-N) : τάση ανάμεσα στην φάση και τον ουδέτερο
2. F : συχνότητα
3. I(L1) : ρεύμα γραμμής
4. P(ΣL) : συνολική ενεργός ισχύς
5. Q(ΣL) : συνολική άεργος ισχύς
6. S(ΣL) : συνολική φαινόμενη ισχύς
7. PF(ΣL) IEC (§) : συνολικός συντελεστής ισχύος
8. Ea+ (ΣT) : συνολική ενέργεια που εισέρχεται στο κύκλωμα
9. Ea- (ΣT) : συνολική ενέργεια που εξέρχεται από το κύκλωμα

Κάθε μέτρηση παραπάνω, αντιστοιχεί και σε ένα καταχωρητή με μοναδική διεύθυνση όπως φαίνονται στον πίνακα της σελίδας 8. Παρατηρούμε ότι δεν έχουν όλες οι μεταβλητές την ίδια ακρίβεια κι αφετέρου ότι δεν αντιστοιχούν στον ίδιο αριθμό θέσεων μνήμης. Για παράδειγμα, η τάση έχει ακρίβεια 0.01 V και άρα μπορεί να μετρήσει τιμές από 92.00 – 276.00 V. Επίσης, αντιστοιχεί σε μία θέση μνήμης 16 bit που έχει διεύθυνση: B000. Το σύμβολο: H'03' δείχνει ότι η τάση βρίσκεται σε θέση μνήμης του

πεδίου των καταχωρητών τύπου holding. Παρατηρούμε ότι κάθε μέτρηση που θα πάρουμε αντιστοιχεί σε 1 ή 2 θέσεις μνήμης του ίδιου πεδίου (holding registers).

V(L1-N)	V	0,01	B000	1	U16	H'03'	9200…27600

Για το ρεύμα γραμμής, από την άλλη μεριά, παρατηρούμε ότι μετράει με ακρίβεια 1 mA και άρα μπορεί να πάρει τιμές από 0 mA - 40.000 mA ή διαφορετικά από 0 A – 40 A. Πράγματι, ο ενεργειακός μετρητής έχει μέγιστη τιμή για το ρεύμα γραμμής τα 40 A. Παρατηρούμε, επίσης, ότι το ρεύμα αποθηκεύεται σε 2 θέσεις μνήμης και 32 bit συνολικά ενώ η διεύθυνση της πρώτης θέσης μνήμης είναι το B009. Η επόμενη θα είναι η B00A.

I(L1)	mA	1	B009	2	U32	H'03'	0…40000

Η ανάγνωση των 2 θέσεων μνήμης προϋποθέτει ότι γνωρίζουμε ποιά τιμή θα αντιστοιχεί στα πρώτα 16 bit (στα αριστερά) και ποιά στα επόμενα (στα δεξιά) σε έναν αριθμό 32 bit. Αυτό είναι ιδιαίτερης σημασίας γιατί μια λάθος αντιστοίχιση θα έχει το αναπόφευκτο αποτέλεσμα της λάθος μέτρησης. Για τον συγκεκριμένο ενεργειακό μετρητή, τα πρώτα 16 bit αντιστοιχούν στην B009 και τα επόμενα 16 bit στην B00A.

Προκειμένου να γίνει ανάγνωση σωστά θα πρέπει να επιτελεστούν πράξεις σε επίπεδο bit που θα δούμε παρακάτω.

Ο ενεργειακός μετρητής μπορεί να δώσει τιμές για διάφορες άλλες μεταβλητές όπως π.χ. ενέργειες ανά κόστος (tariff) αλλά αυτές οι τιμές δεν μας αφορούν μια που τις πράξεις για τον υπολογισμό του συνολικού κόστους της κατανάλωσής μας θα τις κάνουμε μόνοι μας χωρίς τη συμβολή του ενεργειακού μετρητή.

Υπόβαθρο

Κάθε θέση μνήμης σε μια διάταξη modbus εκτός από τη μοναδική διεύθυνση στην οποία αντιστοιχεί, έχει και πληροφορία για το είδος του πεδίου στο οποίο ανήκει. Κάθε modbus σύστημα έχει 4 είδη πεδίου για τις θέσεις μνήμης:

1. Coils : 1bit (θέσεις μνήμης από 1 – 2710 στο 16-δικό σύστημα)
2. Inputs : 1 bit (θέσεις μνήμης από 2711 – 4E20 στο 16-δικό σύστημα)
3. Input Registers : 16 bit (θέσεις μνήμης από 7531 – 9C40 στο 16-δικό σύστημα)
4. Holding Registers : 16 bit (θέσεις μνήμης από 9C41 – C350 στο 16-δικό σύστημα)

Κάθε πεδίο καθορίζει και τον αριθμό των bit που οι θέσεις μνήμης που βρίσκονται σε αυτό μπορούν να αποθηκεύσουν. Για παράδειγμα τα Coils αντιστοιχούν σε θέσεις μνήμης 1 bit όπως και τα inputs. Από την άλλη μεριά, οι καταχωρητές input και holding αντιστοιχούν σε θέσεις μνήμης 16 bit. Προκειμένου να

αποθηκευτούν παραπάνω από 1 bit για τα πεδία coils και inputs και παραπάνω των 16 bit για τα πεδία input registers και holding registers θα απαιτηθούν παραπάνω της 1 θέσεις μνήμης.

Παρατηρήσαμε παραπάνω ότι στη σήμανση του είδους του πεδίου στο οποίο βρίσκονται οι θέσεις μνήμης όπου αποθηκεύονται τα στοιχεία που μετρά ο ενεργειακός μετρητής, ήταν ως εξής: H'03'. Το H’ στην αρχή δηλώνει το πεδίο (holding register) ενώ το 03’ δηλώνει το είδος της συνάρτησης που θα πρέπει το arduino να καλέσει προκειμένου να διαβάσει τα περιεχόμενα των θέσεων μνήμης του ενεργειακού μετρητή. Γενικά, υπάρχουν οι παρακάτω συναρτήσεις για το σύστημα modbus:

1. Read Coil Status : 01
2. Read Input Status : 02
3. Read Input Registers : 04
4. Read Holding Registers : 03
5. Force Single Coil : 05
6. Preset Single Register : 06

Οι 4 πρώτες συναρτήσεις διαβάζουν δεδομένα από τη διάταξη modbus. Οι συναρτήσεις 5 και 6 γράφουν δεδομένα σε αντίστοιχα πεδία της διάταξης. Όπως παρατηρούμε, η συνάρτηση 03’ αντιστοιχεί στην ανάγνωση του πεδίου καταχωρητών holding και άρα είναι αυτή που θα χρησιμοποιήσουμε παρακάτω όταν θα υλοποιήσουμε την επικοινωνία με το arduino. Το ίδιο μας προτείνει και η εταιρεία στο βιβλίο οδηγιών της.

Προκειμένου, όμως, να έχουμε μια εικόνα του τι πρέπει να στείλουμε ως ερώτημα από το arduino στον ενεργειακό μετρητή και τι θα πρέπει να αναμένουμε να πάρουμε, θα δούμε το παρακάτω παράδειγμα. Έστω ότι θέλουμε να διαβάσουμε την τιμή της τάσης και την τιμή του ρεύματος γραμμής από τον ενεργειακό μετρητή. Θα πρέπει να στείλουμε σε αυτόν από το arduino μια σειρά από bits τα οποία και θα καθορίζουν με σαφήνεια:

1. το μοναδικό αναγνωριστικό του ενεργειακού μετρητή (id)
2. το είδος της λειτουργίας που θα θέλαμε να επιτελέσουμε (function code)
3. την αρχική διεύθυνση μνήμης που θα θέλαμε να χρησιμοποιήσουμε (θα πρέπει να είναι συμβατή με τη λειτουργία που θα θέλαμε να επιτελέσουμε παραπάνω)
4. τον αριθμό των θέσεων μνήμης που θα θέλαμε να διαβάσουμε

Για την περίπτωσή μας, ο μετρητής ενέργειας έχει id = 1, η λειτουργία που θα θέλαμε να επιτελέσουμε έχει κωδικό 03, η διεύθυνση μνήμης που θα διαβάσουμε είναι η B000 που, όπως φαίνεται και από το πεδίο, είναι μια θέση μνήμης που αντιστοιχεί στο πεδίο των holding καταχωρητών (16 bit), και ο αριθμός των θέσεων μνήμης που θα χρειαστούμε είναι 1. Συνεπώς, η ερώτηση που θα πρέπει να στείλουμε από το arduino στον ενεργειακό μετρητή θα πρέπει να έχει τη μορφή:

[01] [03] [B0] [00][00][01][X][X]

Η παραπάνω ακολουθία bytes (8 bit) μας λέει:

1. [01] : η διεύθυνση ή το id του ενεργειακού μετρητή
2. [03] : ο τύπος της συνάρτησης που θα χρησιμοποιήσουμε (Read Holding Registers)
3. [B0] : το high (πρώτο) byte της διεύθυνσης στο πεδίο των καταχωρητών holding που θα διαβάσουμε
4. [00] : το low (δεύτερο) byte της διεύθυνση στο πεδίο των καταχωρητών holding που θα διαβάσουμε
5. [00] : το high (πρώτο) byte του αριθμού των διευθύνσεων στο πεδίο των καταχωρητών holding που θα διαβάσουμε
6. [01] : το low (δεύτερο) byte του αριθμού των διευθύνσεων στο πεδίο των καταχωρητών holding που θα διαβάσουμε
7. [X] [X] : ο 16 bit CRC (error checking) κώδικας που θα χρησιμοποιήσουμε (δημιουργείται αυτόματα)

Επειδή ο αριθμός των θέσεων μνήμης κάθε πεδίου είναι 10.000, ο αριθμός των θέσεων μνήμης που θα μπορούσαμε να διαβάσουμε θα μπορούσε να είναι το πολύ 10.000 = 2710 στο 16-δικό σύστημα ([27] [10]). Επίσης, σε κάποιες περιπτώσεις αντί για την καθαρή θέση μνήμης που θα θέλαμε να διαβάσουμε, στέλνουμε την θέση μνήμης σε σχέση με την αρχική θέση του αντίστοιχου πεδίου. Για την περίπτωσή μας θα ήταν η B000 - 9C41 = 13BF. Συνεπώς, η παραπάνω σειρά bytes που θα στείλει το arduino στον ενεργειακό μετρητή θα έχει ως εξής:

[01] [03] [13] [BF][00][01][X][X]

Η απάντηση που θα αναμένουμε από τον ενεργειακό μετρητή για μα τιμή π.χ. 230 V = E6 V (στο 16-δικό σύστημα) της τάσης θα είναι:

[01] [03] [02] [00] [E6] [X][X]

Η παραπάνω ακολουθία bytes (8 bit) σημαίνει:

8. [01] : η διεύθυνση ή το id του ενεργειακού μετρητή
9. [03] : ο τύπος της συνάρτησης που χρησιμοποιήσαμε (Read Holding Registers)
10. [02] : ο αριθμός των byte που ακολουθούν κι αντιστοιχούν στις μετρήσεις
11. [00] : το high (πρώτο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε
12. [E6] : το low (δεύτερο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε
13. [X] [X] : ο 16 bit CRC (error checking) κώδικας που χρησιμοποιήσαμε (δημιουργείται αυτόματα)

Αντίστοιχα, για το ρεύμα και μια τιμή π.χ. 0.23 A = 230 mA = E6 mA, η ερώτηση που θα στείλει το arduino στον ενεργειακό μετρητή θα είναι της μορφής:

[01] [03] [B0] [09][00][02][X][X]

Η παραπάνω ακολουθία bytes (8 bit) μας λέει:

14. [01] : η διεύθυνση ή το id του ενεργειακού μετρητή
15. [03] : ο τύπος της συνάρτησης που θα χρησιμοποιήσουμε (Read Holding Registers)
16. [B0] : το high (πρώτο) byte της διεύθυνσης στο πεδίο των καταχωρητών holding που θα διαβάσουμε
17. [09] : το low (δεύτερο) byte της διεύθυνση στο πεδίο των καταχωρητών holding που θα διαβάσουμε
18. [00] : το high (πρώτο) byte του αριθμού των διευθύνσεων στο πεδίο των καταχωρητών holding που θα διαβάσουμε
19. [02] : το low (δεύτερο) byte του αριθμού των διευθύνσεων στο πεδίο των καταχωρητών holding που θα διαβάσουμε
20. [X] [X] : ο 16 bit CRC (error checking) κώδικας που θα χρησιμοποιήσουμε (δημιουργείται αυτόματα)

Όπως και προηγουμένως, σε κάποιες περιπτώσεις αντί για την καθαρή θέση μνήμης που θα θέλαμε να διαβάσουμε, στέλνουμε την θέση μνήμης σε σχέση με την αρχική θέση του αντίστοιχου πεδίου. Για την περίπτωσή μας θα ήταν η B009 - 9C41 = 13C8. Συνεπώς, η παραπάνω σειρά bytes που θα στείλει το arduino στον ενεργειακό μετρητή θα έχει ως εξής:

[01] [03] [13] [C8][00][02][X][X]

Η απάντηση που θα αναμένουμε από τον ενεργειακό μετρητή για μα τιμή π.χ. 230 mA = E6 mA (στο 16-δικό σύστημα) της τάσης θα είναι:

[01] [03] [04] [00] [00] [00] [E6] [X][X]

Η παραπάνω ακολουθία bytes (8 bit) σημαίνει:

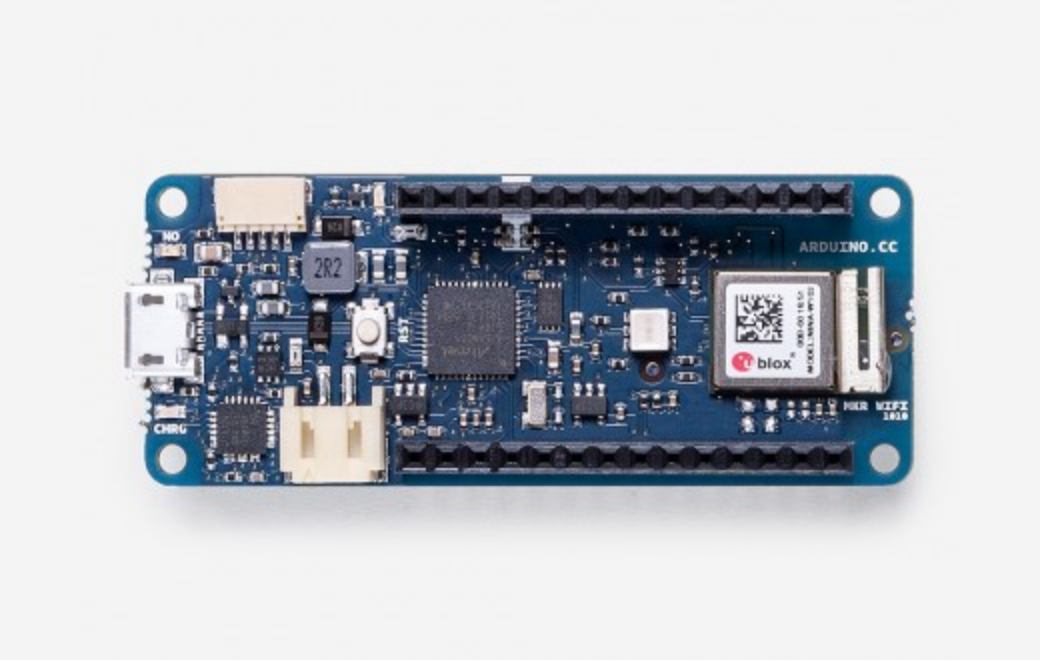
21. [01] : η διεύθυνση ή το id του ενεργειακού μετρητή
22. [03] : ο τύπος της συνάρτησης που χρησιμοποιήσαμε (Read Holding Registers)
23. [04] : ο αριθμός των byte που ακολουθούν κι αντιστοιχούν στις μετρήσεις
24. [00] : το high (πρώτο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε
25. [00] : το low (δεύτερο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε
26. [00] : το high (πρώτο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε
27. [E6] : το low (δεύτερο) byte της τιμής της θέσης μνήμης στο πεδίο των καταχωρητών holding που διαβάσαμε

28. [X] [X] : ο 16 bit CRC (error checking) κώδικας που χρησιμοποιήσαμε (δημιουργείται αυτόματα)

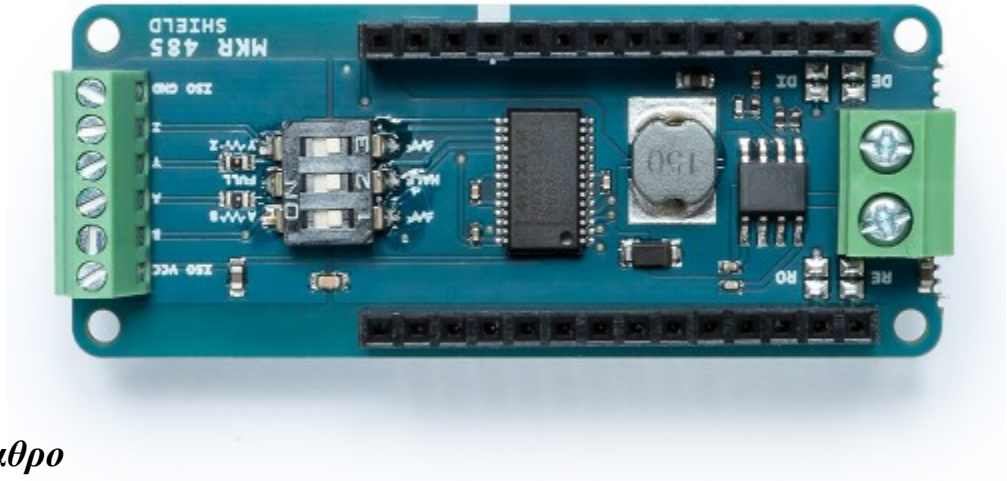
Οι τιμές για το ρεύμα από 0 – 999 mA καταχωρούνται στην θέση μνήμης B009 ενώ από εκεί και πάνω διαχωρίζονται και μια τιμή π.χ. 1024 mA θα καταχωρηθεί ως εξής: [00] [01] [00] [18] = 1 A και 0.024 A. Τα πρώτα 2 byte θα καταχωρηθούν στη θέση μνήμης B00A και τα δύο τελευταία byte στη θέση μνήμης B009. Με βάση αυτό το σκεπτικό αναπτύξαμε το πρόγραμμα σε arduino και διαβάσαμε τις τιμές του ενεργειακού μετρητή.

Arduino

Το Arduino είναι μια διάταξη που περιλαμβάνει έναν μικροελεγκτή και εισόδους – εξόδους προκειμένου να συνδεθεί με άλλες διατάξεις. Η γλώσσα προγραμματισμού του Arduino βασίζεται στην γλώσσα C ενώ το περιβάλλον ανάπτυξης κώδικα για Arduino είναι γραμμένο σε Java. Το arduino περιλαμβάνει πολλές διαφορετικές διατάξεις που επιτελούν διαφορετικές λειτουργίες. Για την περίπτωσή μας, χρησιμοποιήσαμε την διάταξη Arduino MKR WiFi 1010 η οποία και αποτελεί τη βάση πάνω στην οποία συνδέσαμε το Arduino MKR 485 Shield το οποίο συνδέθηκε μέσω RS-485 με τον ενεργειακό μετρητή. Η σύνδεση επιτελέστηκε με τη βοήθεια απλού πλακέ τηλεφωνικού καλώδιου.



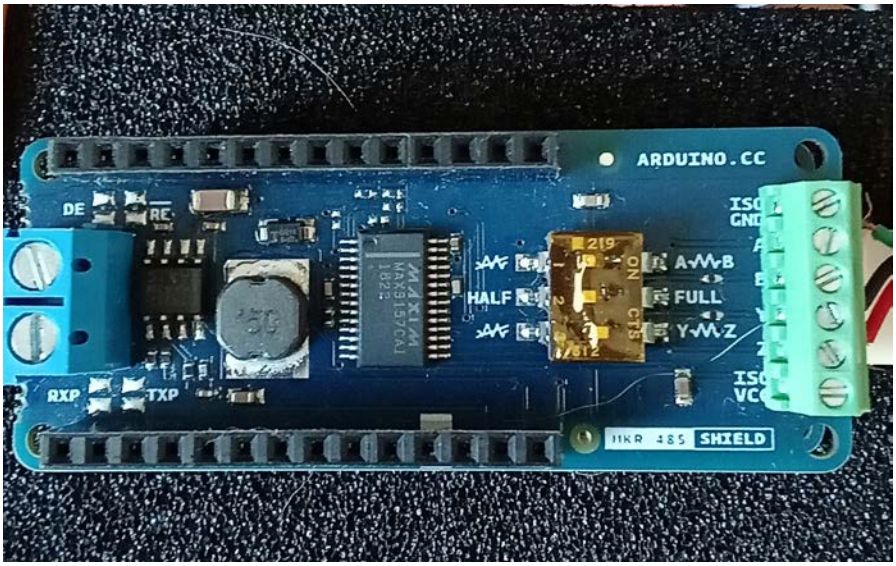
Η τροφοδοσία του Arduino MKR WiFi 1010 έγινε μέσω φορτιστή κινητού τηλεφώνου με βίσμα USB-C. Η πλακέτα Arduino MKR 485 Shield δεν απαιτούσε ξεχωριστή τροφοδοσία.



Υπόβαθρο

Στοιχεία Διάταξης

Προκειμένου να συνδέσουμε τον ενεργειακό μετρητή με τις ρυθμίσεις που έχουμε κάνει αναφορικά με την ταχύτητα της γραμμής, τον αριθμό των data και stop bits αλλά και το είδος της ισοτιμίας που θα χρησιμοποιήσουμε, θα πρέπει, αντίστοιχα, να καθορίσουμε στις ίδιες τιμές τα αντίστοιχα μεγέθη στο arduino. Για να γίνει αυτό θα χρειαστούμε αφενός να τοποθετήσουμε στις σωστές θέσεις τους 3 διακόπτες που βρίσκονται στην πλακέτα Arduino MKR 485 Shield.



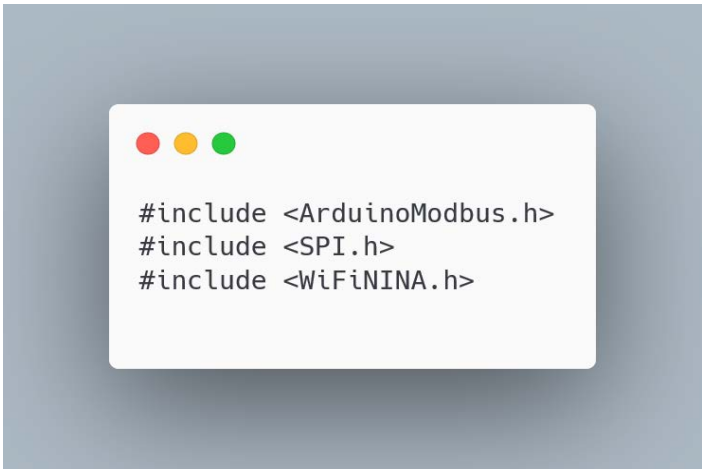
Όπως φαίνεται από την παραπάνω εικόνα, μια που θα επικοινωνούμε μέσω half duplex σύνδεσης, θα πρέπει να απενεργοποιήσουμε την full duplex (ο μεσαίος διακόπτης). Επειδή, επίσης, θα χρησιμοποιήσουμε τα κανάλια Y και Z, θα πρέπει να απενεργοποιήσουμε τον πρώτο (πάνω) διακόπτη και να ενεργοποιήσουμε τον τελευταίο (κάτω). Η ενεργοποίηση των Y – Z καναλιών γίνεται γιατί η half duplex επικοινωνία συμβαίνει αποκλειστικά ανάμεσα σε αυτά τα κανάλια. Οπότε και η σύνδεση των αντίστοιχων καλωδίων είναι ξεκάθαρη. Θα χρησιμοποιήσουμε τις θέσεις Y, Z και ISOGND στην κλέμμα.

Όπως φαίνεται από την παραπάνω εικόνα, το κόκκινο καλώδιο συνδέθηκε στην υποδοχή Z, το μαύρο στην υποδοχή Y ενώ το πράσινο στο ISOGND. Αντίστοιχα, θα πρέπει στον ενεργειακό μετρητή να συνδέσουμε το κόκκινο στο (-) το μαύρο στο (+) και το πράσινο στην υποδοχή (0).

Στην παραπάνω εικόνα φαίνεται και η σύνδεση της τερματικής αντίστασης SMC120R των 120 Ohm ανάμεσα στους ακροδέκτες (+) και (-) του RS-485. Αν και δεν είναι απαραίτητη η αντίσταση αυτή στην περίπτωσή μας μια και έχουμε μόνο μια διάταξη modbus (ο μετρητής ενέργειας) την οποία διαβάζουμε, αποτελεί σημαντική προσθήκη σε περίπτωση που αργότερα αποκτήσουμε κι άλλες συσκευές modbus που θα θέλαμε να εντάξουμε στο δίκτυο επικοινωνίας μια που θα ελαχιστοποιήσει τον θόρυβο.

Στοιχεία Εφαρμογής

Αρχικά θα πρέπει να φορτώσουμε τις αντίστοιχες βιβλιοθήκες στην εφαρμογή Arduino προκειμένου να διαβάσουμε τα στοιχεία από τον ενεργειακό μετρητή μέσω modbus.



Η βιβλιοθήκη για την επικοινωνία μέσω modbus είναι η ArduinoModbus.h και η SPI.h και αφορούν στο Arduino MKR 485 Shield ενώ η άλλη βιβλιοθήκη είναι για την επικοινωνία του Arduino MKR WiFi 1010 με το internet. Η βιβλιοθήκη ArduinoModbus.h περιέχει τις συναρτήσεις ανάγνωσης που θα χρησιμοποιήσουμε ενώ η SPI περιέχει τις συναρτήσεις που χρειαζόμαστε ώστε το arduino να γίνει ο master και η ενεργειακός μετρητής η διάταξη slave όπως απαιτείται από το modbus πρωτόκολλο.

Για την ανάγνωση π.χ. της τάσης από τον ενεργειακό μερτητή θα χρησιμοποιήσουμε την συνάρτηση readVoltage() όπου θέτουμε ως 0x01 τη διεύθυνση (id) του ενεργειακού μετρητή, ως HOLDING_REGISTERS το είδος του πεδίου στο οποίο βρίσκονται οι θέσεις μνήμης που θα διαβάσουμε, ως 0xB000 τη θέση μνήμης που θα διαβαστεί και ως 0x01 τον αριθμό των συνεχόμενων θέσεων μνήμης που θα διαβαστούν.

Επειδή η θέση μνήμης είναι 1 και περιέχει 16 bit πληροφορία, την αποθηκεύουμε σε έναν ακέραιο χωρίς πρόσημο 16 bit και τη διαιρούμε με το 100 προκειμένου να τη μετατρέψουμε σε Volt με ακρίβεια 0.01 V. Εάν δεν διαβαστεί κάτι από τη θέση αυτή ή εάν η θέση μνήμης δεν είναι διαθέσιμη η συνάρτηση θα επιστρέψει την τιμή 0.00 V για την τάση.

Αντίστοιχα, για την ανάγνωση της τιμής του ρεύματος από τον ενεργειακό μετρητή θα χρησιμοποιήσουμε την συνάρτηση readCurrent() όπου θέτουμε ως 0x01 τη διεύθυνση (id) του ενεργειακού μετρητή, ως HOLDING_REGISTERS το είδος του πεδίου στο οποίο βρίσκονται οι θέσεις μνήμης που θα διαβάσουμε, ως 0xB009 τη θέση μνήμης που θα διαβαστεί πρώτη και ως 0x02 τον αριθμό των συνεχόμενων θέσεων μνήμης που θα διαβαστούν.

```
float readVoltage(){
  float volt = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB000: The HEX address of the Voltage Data (V(L1-N))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, voltage is kept in 1
register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB000, 0x01)) { //make the call to the
register
    //Serial.print("Failed to read voltage! ");
    //Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we need to divide by 100 to get the
      * actual value in Volt
     *
     */
    volt = register_data/100.0;
  }
  return volt;
}
```

Επειδή οι θέσεις μνήμης είναι 2 και περιέχουν 32 bit πληροφορία συνολικά, την αποθηκεύουμε σε δύο ακέραιους χωρίς πρόσημο 32 bit. Μετά, μεταφέρουμε τον πρώτο ακέραιο κατά 16 bit στα αριστερά και τον προσθέτουμε στον δεύτερο ακέραιο ώστε να πάρουμε το συνολικό αποτέλεσμα σε μια τιμή 32 bit. Την τιμή αυτή, μετά, τη διαιρούμε με το 1000 προκειμένου να τη μετατρέψουμε σε Ampere με ακρίβεια 0.001 A. Εάν δεν διαβαστεί κάτι από τις θέσεις αυτές ή εάν οι θέσεις μνήμης δεν είναι διαθέσιμες η συνάρτηση θα επιστρέψει την τιμή 0.000 A για το ρεύμα.

Αντίστοιχα, διαβάζουμε και τις υπόλοιπες μεταβλητές χρησιμοποιώντας τις θέσεις μνήμης που δίνονται στον πίνακα της σελίδας 8.

Προκειμένου, όμως, να επιτύχουμε σωστή επικοινωνία θα πρέπει να δηλώσουμε στο Arduino ότι επιθυμούμε επικοινωνία 9600 bps με 8 data, 1 stop και καθόλου ισοτιμία. Αυτό γίνεται με την εκκίνηση του ModbusRTUClient στο arduino όπου τίθεται η τιμή των 9600 bps ενώ έχει προκαθοριστεί η χρήση 8 data, 1 stop και καθόλου ισοτιμίας μέσω του SERIAL_8N1 που το arduino υποθέτει ότι ισχύει σε περίπτωση που στην εντολή ModbusRTUClient.begin δεν δοθεί κάτι άλλο από τις παρακάτω επιλογές:

1. SERIAL_5N1

2. SERIAL_6N1
3. SERIAL_7N1
4. SERIAL_8N1 (η προκαθορισμένη ρύθμιση)
5. SERIAL_5N2
6. SERIAL_6N2
7. SERIAL_7N2
8. SERIAL_8N2
9. SERIAL_5E1: άρτια ισοτιμία
10. SERIAL_6E1
11. SERIAL_7E1
12. SERIAL_8E1
13. SERIAL_5E2
14. SERIAL_6E2
15. SERIAL_7E2
16. SERIAL_8E2
17. SERIAL_5O1: περιττή ισοτιμία
18. SERIAL_6O1
19. SERIAL_7O1
20. SERIAL_8O1
21. SERIAL_5O2
22. SERIAL_6O2
23. SERIAL_7O2
24. SERIAL_8O2

```
if (!ModbusRTUClient.begin(9600)) {
   //Serial.println("Failed to start Modbus RTU Client!");
   //do not continue if failed
   while (1);
 }
```

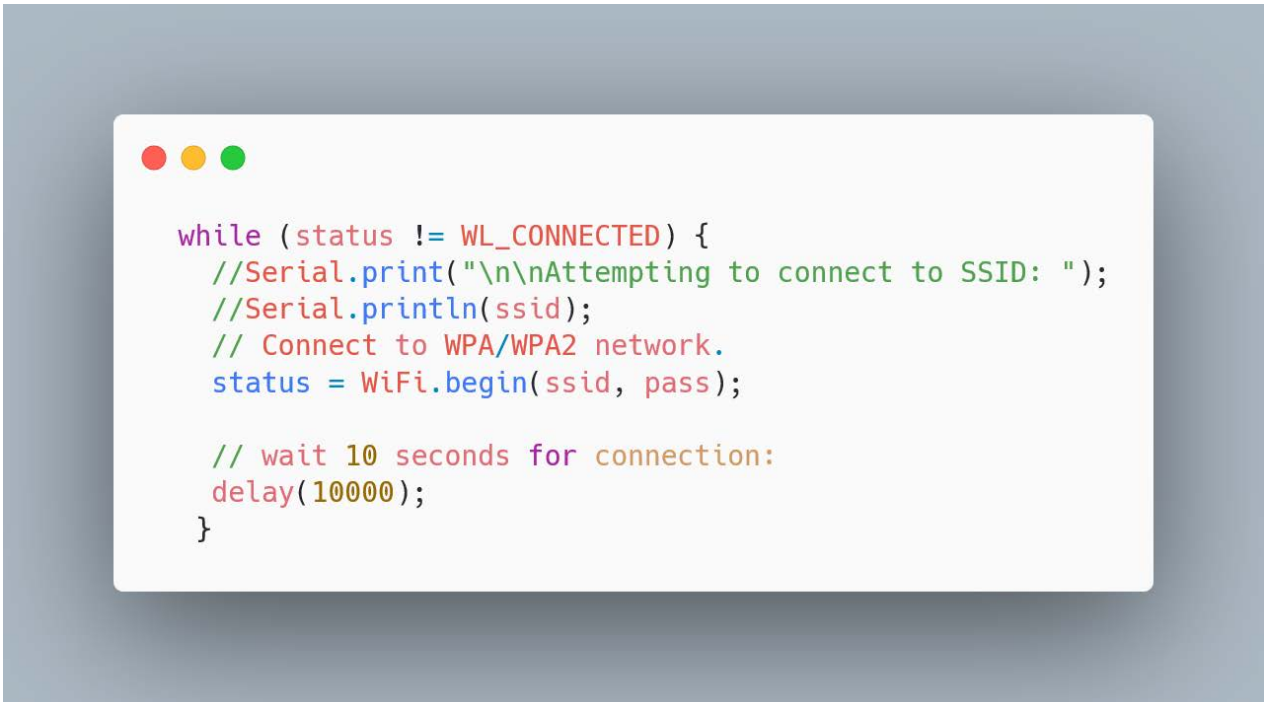
```
float readCurrent(){
  float current = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB009: The HEX address of the Current Data (I(L1))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, current is kept in 2
registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB009, 0x02)) { //make the call to the
register
    //Serial.print("Failed to read current! ");
    //Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    uint32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (U16)
    uint32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (U16)
    uint32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two U16 to form a
U32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the current in mA. To get it in A, we need to divide by 1000.0.
     *
     */
    current = register_data/1000.0;
  }
  return current;
}
```

Όλος ο κώδικας του προγράμματος που χρησιμοποιήσαμε και εγκαταστήσαμε στο arduino προκειμένου να διαβάσουμε τις τιμές του ενεργειακού μετρητή δίνεται στο παράρτημα στο τέλος της εργασίας.

Η επικοινωνία με το wifi γίνεται μέσω του WiFiClient. Αρχικά ελέγχουμε εάν υπάρχει διαθέσιμο κάποιο WiFi υλικό το οποίο θα επιτύχει τη σύνδεση με το δίκτυο :

```
// check for the WiFi module:
 if (WiFi.status() == WL_NO_MODULE) {
   //Serial.println("Communication with WiFi module failed!");
   // don't continue
   while (true);
 }
```

Έπειτα ελέγχουμε εάν έχουμε συνδεθεί στο αντίστοιχο δίκτυο με όνομα αυτό που δίνεται στη μεταβλητή



```
while (status != WL_CONNECTED) {
  //Serial.print("\n\nAttempting to connect to SSID: ");
  //Serial.println(ssid);
  // Connect to WPA/WPA2 network.
  status = WiFi.begin(ssid, pass);

  // wait 10 seconds for connection:
  delay(10000);
 }
```

ssid και password αυτό που δίνεται στην μεταβλητή pass

Εφόσον έχουμε συνδεθεί στο WiFi είμαστε έτοιμοι να στείλουμε τα δεδομένα του ενεργειακού μετρητή στον Server της επιλογής μας. Στην περίπτωσή μας το στέλνουμε εκεί που έχουμε εγκαταστήσει τη βάση δεδομένων.

```
// if you get a connection, report back via serial:
if (client.connect(server, 80)) {
  //Serial.println("Connected to server");
  delay(100);
  // Make a HTTP request:
  String myString = "GET /dissertation/senddata_arduino.php?email=" + username + "&password=" +
password + "&active_energy_in=" + active_energy_in + "&active_energy_out=" + active_energy_out +
"&active_power=" + active_power +"&voltage="+ voltage +"&current="+current+"&frequency="+frequency+"&
power_factor="+power_factor+" HTTP/1.1";
  client.println(myString);
  client.println("Host: moijoune.net");
  client.println("Connection: close");
  client.println();
}
```

Αυτή η διαδικασία – ανάγνωση δεδομένων από τον ενεργειακό μετρητή κι αποστολή στον server – επαναλαμβάνεται κάθε 10 λεπτά με την εντολή delay(600000) (600000 ms = 10 min).

Android

Κάθε διαδικτυακή εφαρμογή έχει αξία όταν μπορεί να είναι προσβάσιμη από οποιοδήποτε σημείο βρίσκεται ο χρήστης και έχει πρόσβαση στο internet. Τη σύγχρονη εποχή, μια από τις βασικές μεθόδους πρόσβασης στο διαδίκτυο είναι μέσω κινητού τηλεφώνου. Αλλά, η παροχή απλά της ιστοσελίδας και η απαίτηση, μετά, από τον χρήστη να εισέρχεται σε αυτή από τον φυλλομετρητή του κινητού του τηλεφώνου δεν ανταποκρίνεται στις απαιτήσεις της σύγχρονης εποχής που υπαγορεύουν την όσο το δυνατόν αμεσότερη πρόσβαση στην πληροφορία. Για το λόγο αυτό αναπτύχθηκε μια εφαρμογή για κινητά με λειτουργικό Android η οποία άμεσα, αξιόπιστα και λεπτομερώς, δίνει τη δυνατότητα στον χρήστη να αντλεί πληροφορίες για την κατάσταση της γραμμής στην οποία βρίσκεται ο ενεργειακός μετρητής.

Η επιλογή του λειτουργικού Android για την ανάπτυξη της εφαρμογής που θα ενημερώνει τον χρήστη για τις τιμές που μετρά ο ενεργειακός μετρητής, έχει, κυρίως, να κάνει με την δημοτικότητα του εν λόγω λειτουργικού και την μεγάλη του διείσδυση στην αγορά. Σύμφωνα με δεδομένα για τον Αύγουστο 2020, το 74.25 % των κινητών τηλεφώνων παγκοσμίως λειτουργούν με android. Η υπερίσχυσή του είναι συντριπτική.

Υπόβαθρο

Ο σχεδιασμός της εφαρμογής είναι ένα πολύ σημαντικό στάδιο στην όλη διαδικασία της ανάπτυξης, μια που αυτός θα καθορίσει όχι μόνο το τι θα απαιτηθεί όσον αφορά στις διάφορες λειτουργίες που θα είναι διαθέσιμες στον χρήστη αλλά και θα καθορίσει και το περιβάλλον – την όψη της εφαρμογής.

Αρχικά, θεωρήθηκε αρκετή η απλή καταγραφή των στοιχείων που έχουν αποθηκευτεί στη διαδικτυακή βάση δεδομένων. Αυτό, όμως, εγκαταλείφθηκε μια που τα δεδομένα αυξάνονται κατά 144 κάθε μέρα και συνεπώς η λίστα με τα στοιχεία πολύ γρήγορα θα ήταν τόσο μεγάλη ώστε να απαιτούσε πολύ μεγάλη υπολογιστική ισχύ από το κινητό τηλέφωνο προκειμένου να λειτουργήσει. Και μια εφαρμογή android δεν θα πρέπει να απασχολεί το λειτουργικό πολύ και, κυρίως, δεν θα πρέπει να χρησιμοποιεί όλους τους διαθέσιμους πόρους με κίνδυνο να τινάξει την λειτουργία του κινητού τηλεφώνου στον αέρα. Σε γενικές γραμμές, τα βασικά χαρακτηριστικά της εφαρμογής θα πρέπει να είναι:

1. Απλή στη χρήση
2. Να παρέχει άμεση πληροφόρηση
3. Να χρησιμοποιεί τους ελάχιστους υπολογιστικούς πόρους
4. Να έχει μεγάλη ταχύτητα και να μην εμποδίζει την λειτουργία άλλων εφαρμογών
5. Να έχει ευχάριστο και σαφές περιβάλλον χρήστη
6. Να γίνεται εύκολα αντιληπτή στον χρήστη
7. Να δίνει όλες τις απαραίτητες πληροφορίες
8. Να μπορεί να αντιμετωπίσει προβλήματα προσβασιμότητας ή/και ταχύτητας δικτύου

Σύμφωνα με τα παραπάνω αλλά και με την απαίτηση να δίνει στον χρήστη όχι μόνο τις πληροφορίες που συλλέγει ο ενεργειακός μετρητής αλλά και πληροφορίες κόστους ανάλογα με την τιμή της κιλοβατώρας όπως την εισάγει ο χρήστης στην εφαρμογή αναπτύξαμε ένα σύστημα που όχι μόνο δίνει τη δυνατότητα στον χρήστη να βρει γρήγορα και εύκολα τις πληροφορίες του ενεργειακού μετρητή που θα ήθελε, αλλά παρέχει στατιστικά στοιχεία χρήσης ανά μήνα καθώς και πληροφορίες κόστους. Γενικά, τα στοιχεία της εφαρμογής που αναπτύξαμε είναι:

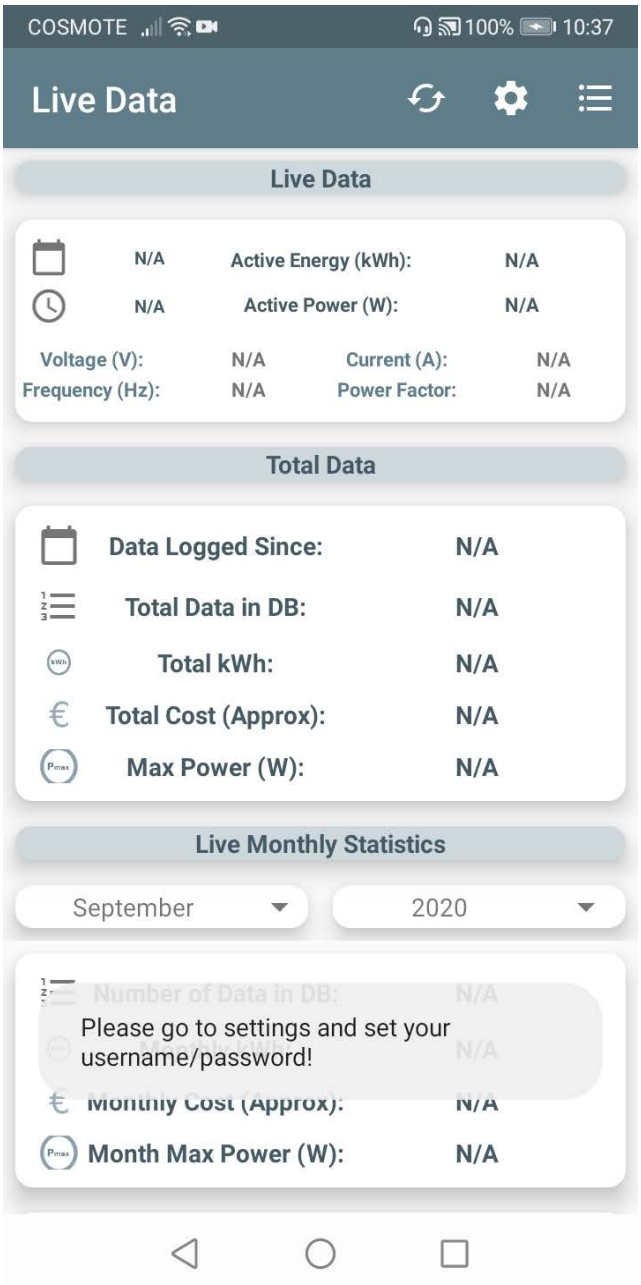
1. Άμεση πληροφόρηση για την εκάστοτε μέτρηση του ενεργειακού μετρητή
2. Συνολικά στοιχεία για τις συνολικές κιλοβατώρες που καταναλώθηκαν, το κόστος τους σύμφωνα με την τιμή της κιλοβατώρας που ρύθμισε ο χρήστης και την μέγιστη ισχύ που καταναλώθηκε στη γραμμή συνολικά
3. Στοιχεία κατανάλωσης και ανά μήνα καθώς και μέγιστα – ελάχιστα για το ρεύμα γραμμής, τάση φάσης – ουδέτερου, συχνότητας εναλλασσόμενου ρεύματος και συντελεστή ισχύος
4. Αναλυτική καταγραφή των στοιχείων του μετρητή ενέργειας ανά ημέρα – μήνα – έτος και δυνατότητα φιλτραρίσματος των στοιχείων αυτών σύμφωνα με τις λέξεις – κλειδιά του χρήστη
5. Αντιμετώπιση προβλημάτων προσβασιμότητας ή / και διαθεσιμότητας σύνδεσης με τη χρήστη κατάλληλων συναρτήσεων ώστε να αποτρέπεται ο εγκλωβισμός της εφαρμογής σε ατέρμονους κύκλους
6. Δυνατότητα χρήσης της εφαρμογής από διαφορετικούς χρήστες

Οι κώδικες που αναπτύχθηκαν δίνονται στο τέλος της εργασίας ενώ την εφαρμογή μπορεί κανείς αν θέλει να την κατεβάσει και να την εγκαταστήσει στο κινητό του από τον σύνδεσμο: http://www.moijoune.net/apks/app-release-ptyxiakh-v6.apk

Στοιχεία Εφαρμογής

Κατά την πρώτη χρήση της εφαρμογής, ο χρήστης θα πρέπει να δώσει τα στοιχεία πρόσβασης στην πλατφόρμα καθώς και την τιμή της κιλοβατώρας προκειμένου να γίνουν οι υπολογισμοί. Για την τιμή της κιλοβατώρας υπάρχει προκαθορισμένη η τιμή 0,09460 €/kWh.

Η παροχή των στοιχείων πρόσβασης είναι απαραίτητη γιατί διαφορετικά η εφαρμογή δεν θα μπορεί να συνδεθεί πουθενά με συνέπεια τη μη καταγραφή στοιχείων. Στην περίπτωση που ο χρήστης δεν δώσει στοιχεία πρόσβασης ή τα στοιχεία δεν είναι σωστά, τότε λαμβάνει και το αντίστοιχο μήνυμα. Προσπαθήσαμε, ώστε να ελαχιστοποιήσουμε τις περιπτώσεις όπου ο χρήστης θα δώσει κάτι που θα είναι αντικανονικό σε κάποιο πεδίο με συνέπεια τη λάθος λειτουργία της εφαρμογής.

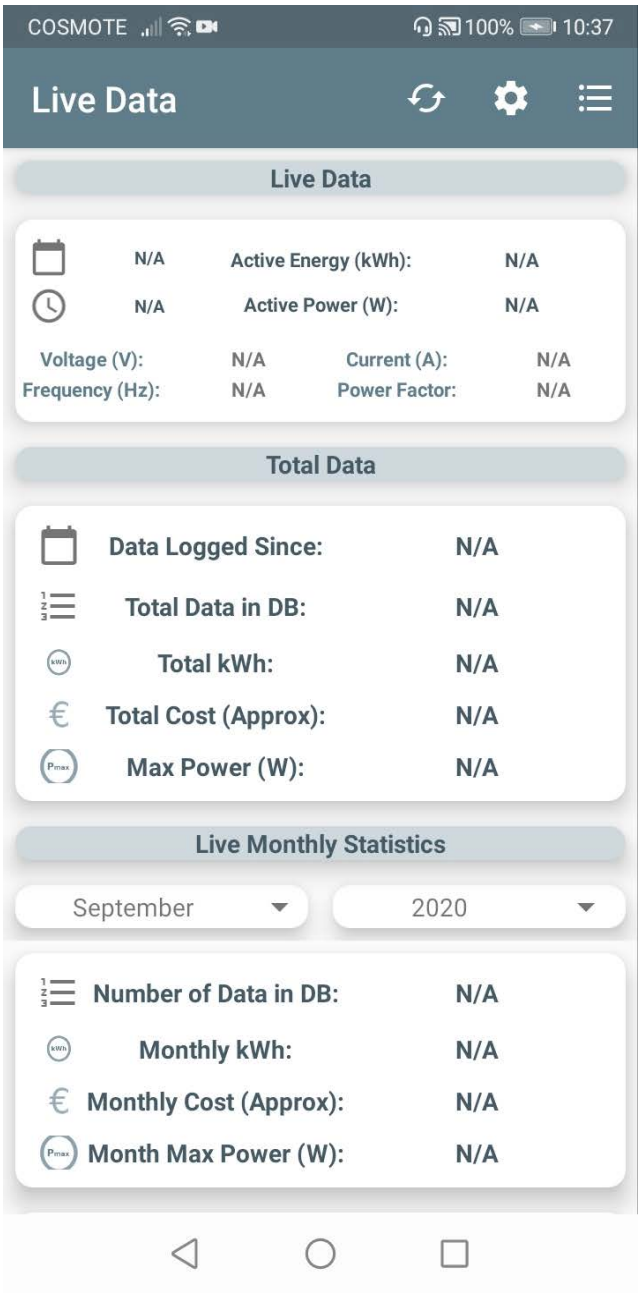


Η επιλογή π.χ. της εμφάνισης των στοιχείων για κάποια συγκεκριμένη ημερομηνία είναι απενεργοποιημένη σε περίπτωση που ο χρήστης δεν δώσει στοιχεία πρόσβασης.

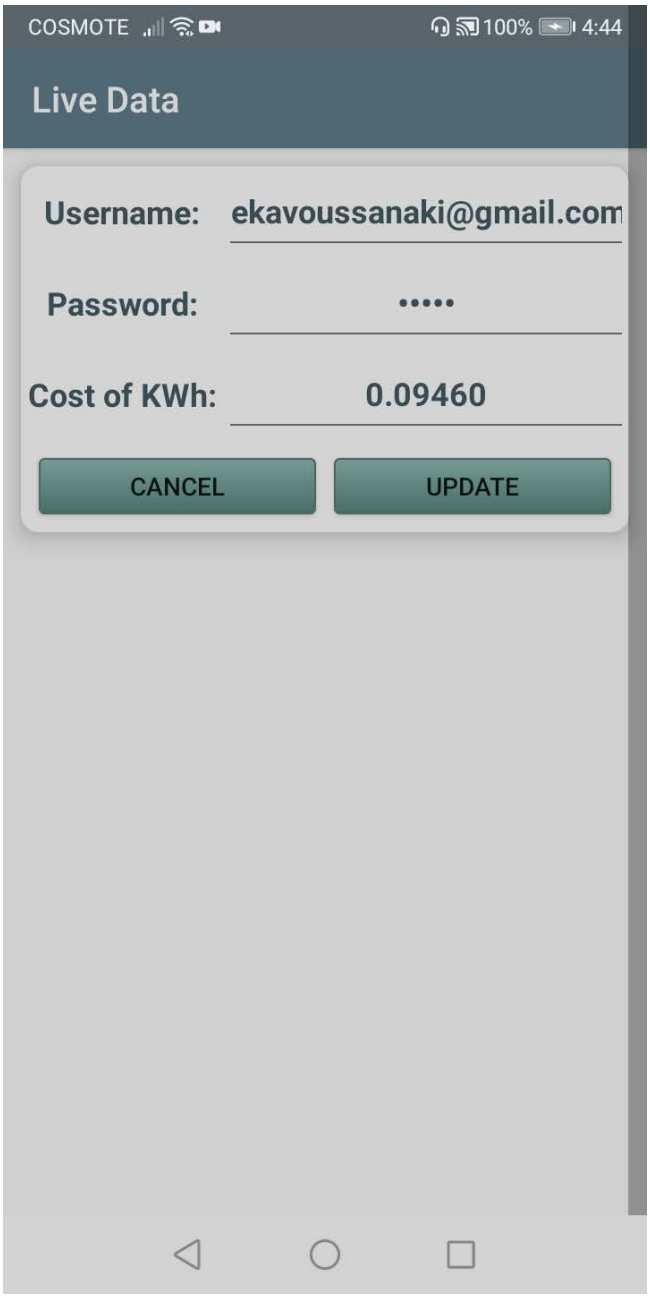
Εφόσον ο χρήστης δεν δώσει στοιχεία πρόσβασης η καρτέλα που θα βλέπει όταν εκκινεί την εφαρμογή θα είναι η παρακάτω με όλα τα πεδία να αναγράφουν N/A (not available).

Παρατηρούμε ότι η εφαρμογή χωρίζεται σε 3 μέρη:

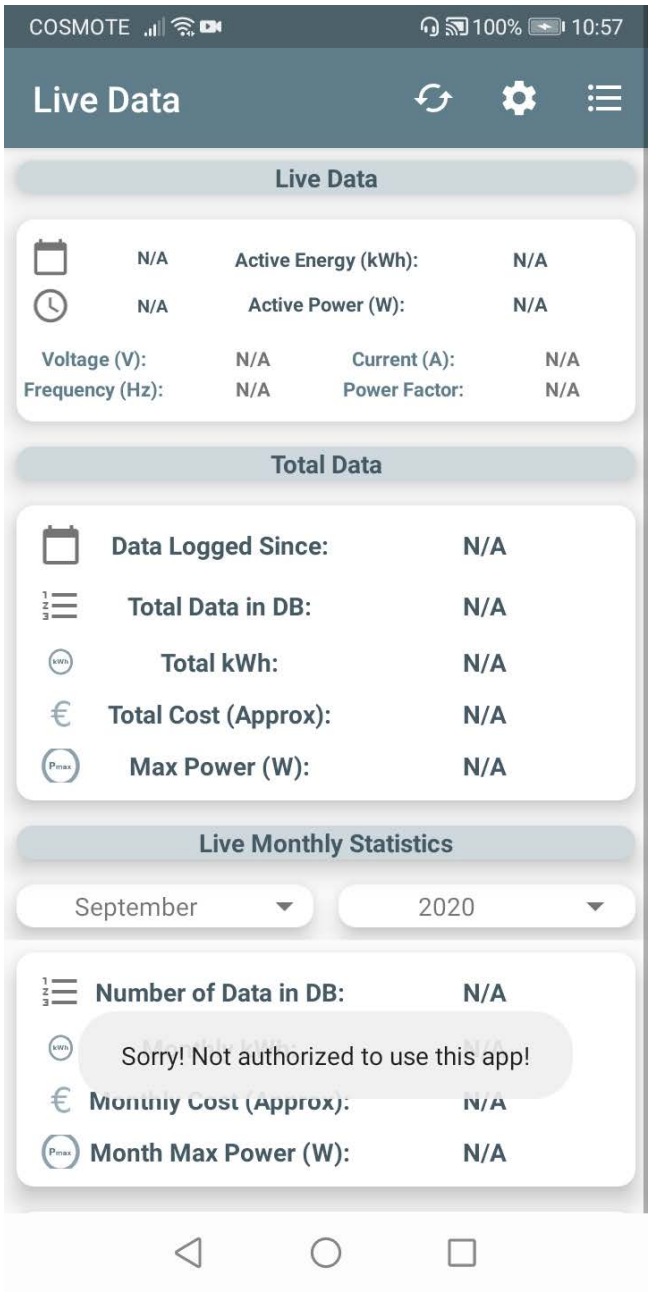
1. Live Data: όπου εμφανίζονται τα πιο πρόσφατα στοιχεία που έχει στείλει ο ενεργειακός μετρητής
2. Total Data: όπου δίνονται συγκεντρωτικές πληροφορίες για τα στοιχεία που έχει έως τώρα συλλέξει ο ενεργειακός μετρητής
3. Live Monthly Statistics: όπου δίνονται πληροφορίες για την κατανάλωση και το κόστος της ανά μήνα καθώς και πληροφορίες για τα μέγιστα – ελάχιστα βασικών μεγεθών όπως η τάση, το ρεύμα, η συχνότητα και ο συντελεστής ισχύος

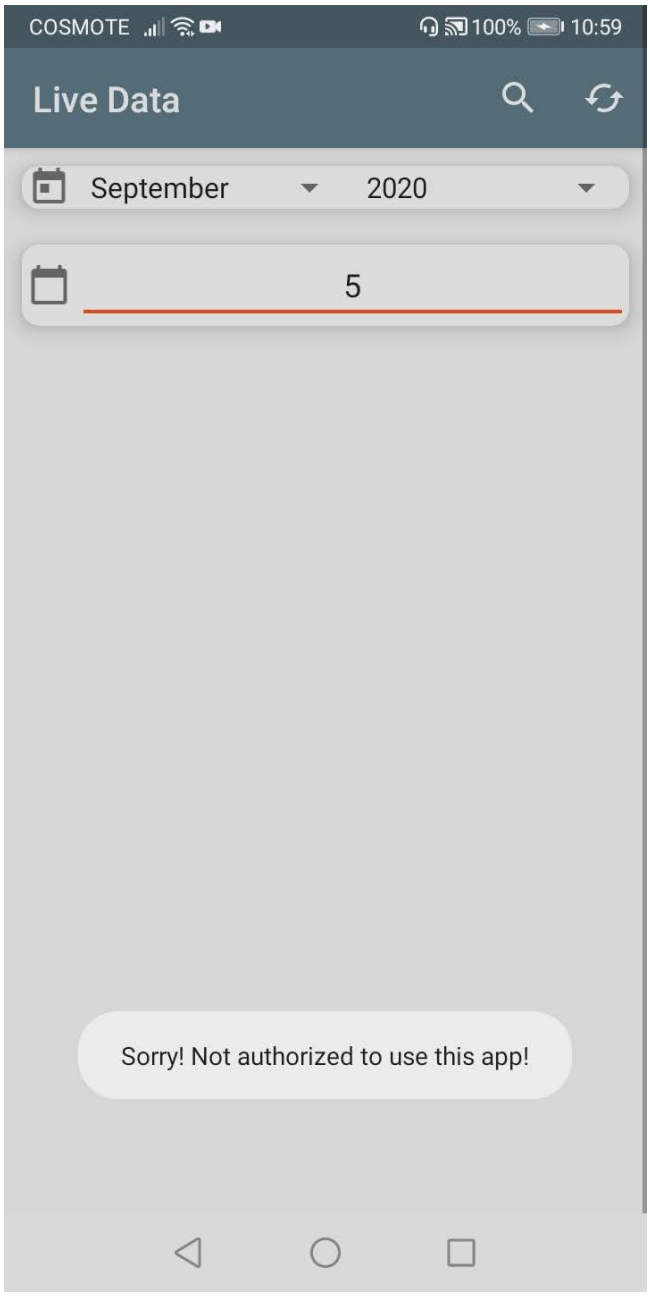


Προκειμένου ο χρήστης να μπορέσει να δει τα δεδομένα του ενεργειακού μετρητή θα πρέπει να εισέλθει στις ρυθμίσεις πατώντας το κουμπάκι με το γρανάζι και να εισάγει τα στοιχεία πρόσβασής του στην πλατφόρμα (όνομα χρήστη (username) – συνθηματικό (password)) ενώ μπορεί να αλλάξει, αν θέλει, την τιμή σύμφωνα με την οποία υπολογίζεται το κόστος της κατανάλωσής του.



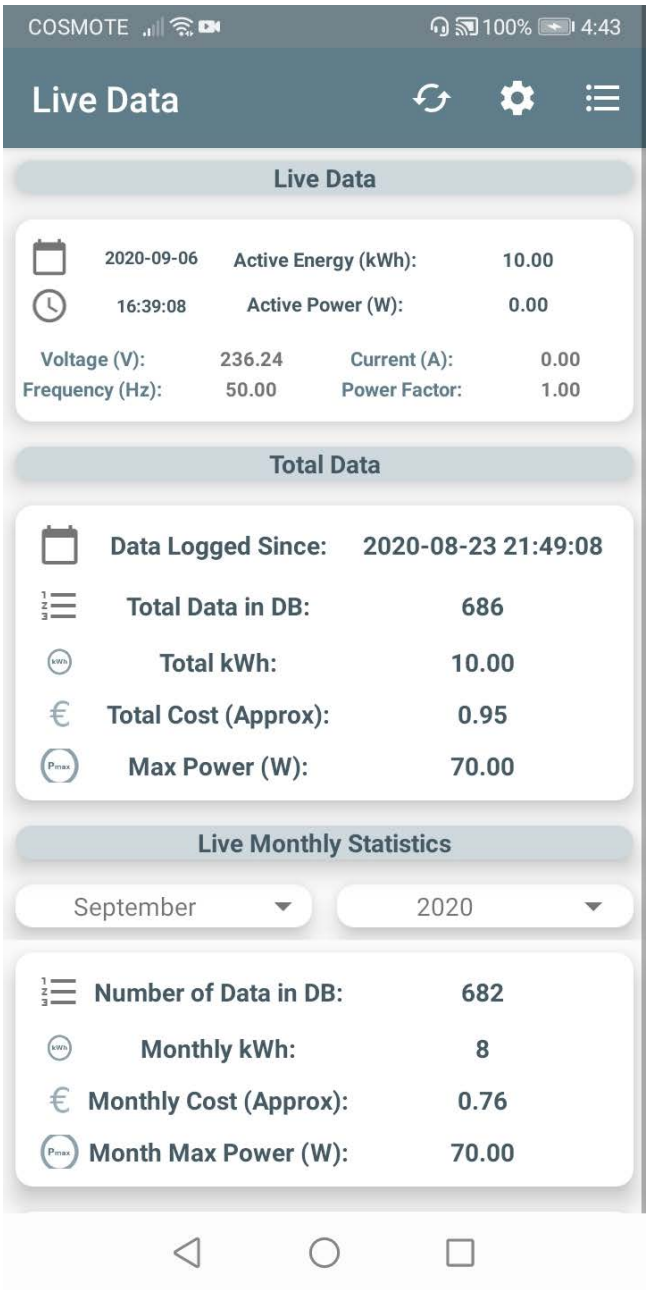
Σε περίπτωση, όπως είπαμε, που τα στοιχεία που δίνει ο χρήστης δεν είναι σωστά παίρνει και το αντίστοιχο μήνυμα όπως φαίνεται παρακάτω:

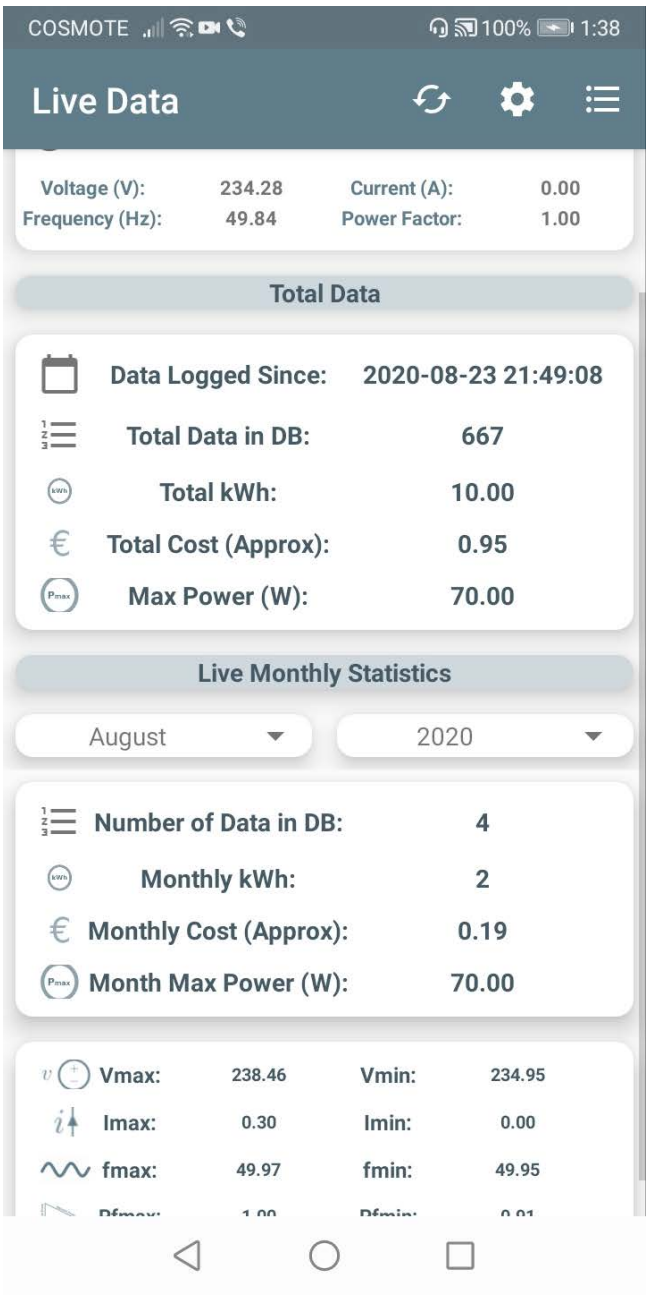




Το ίδιο θα συμβεί κι αν προσπαθήσει να δει στοιχεία για κάποια συγκεκριμένη ημερομηνία. Σε περίπτωση που ο χρήστης έχει ξεχάσει τους κωδικούς πρόσβασής του στην πλατφόρμα θα πρέπει να εισέλθει σε αυτή και να τους ανακτήσει (www.moijoune.net/dissertation). Διαφορετικά, η χρήση της εφαρμογής είναι εντελώς ανώφελη.

Σε περίπτωση που ο χρήστης έχει ανακτήσει τους κωδικούς του και μπορεί να εισέρχεται στην πλατφόρμα, μπορεί να τους εισάγει και στην εφαρμογή προκειμένου να μπορεί να βλέπει τα στοιχεία του ενεργειακού μετρητή και από το κινητό του τηλέφωνο. Εφόσον αυτό έγινε σωστά, πατώντας το κουμπί της ανανέωσης, ο χρήστης θα πάρει μια οθόνη όπως η παρακάτω:

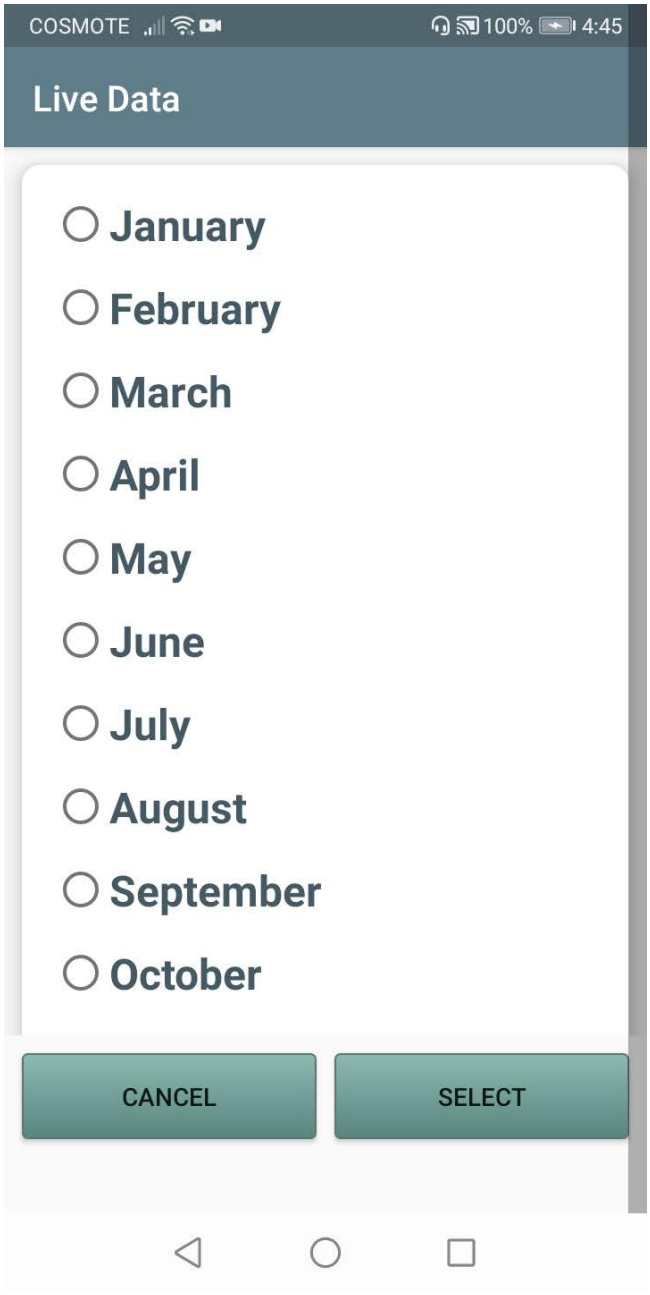


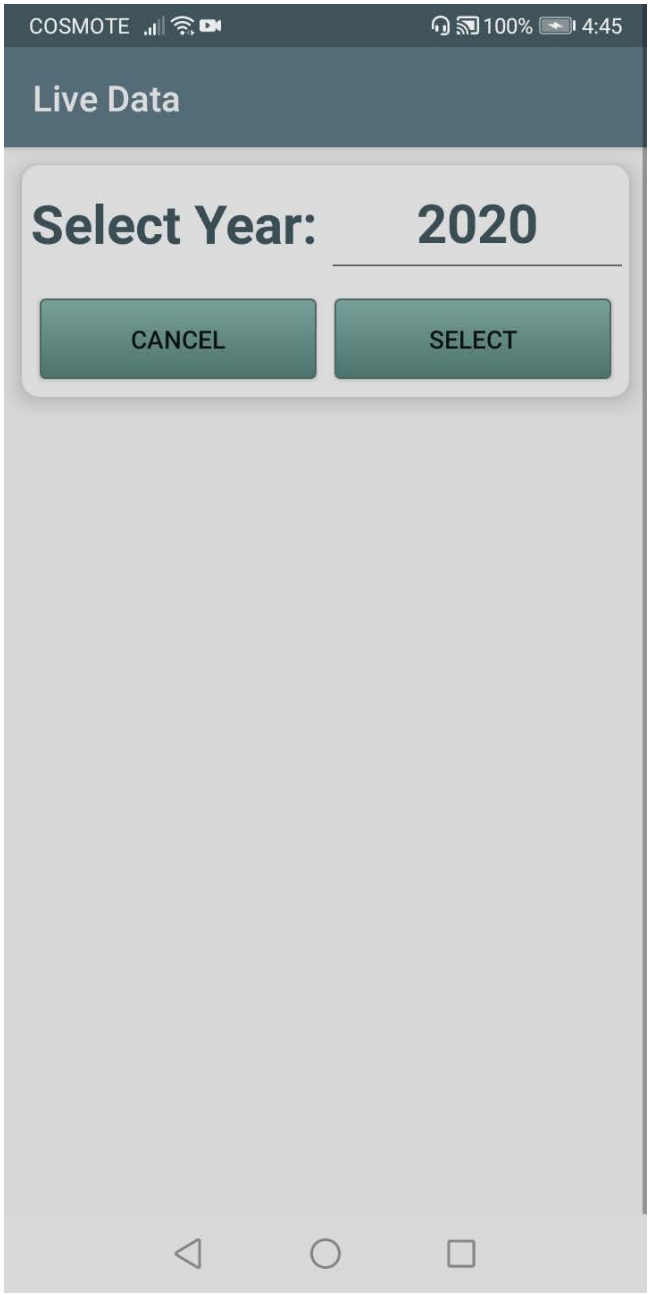


Κάνοντας scroll στην οθόνη, μπορεί ο χρήστης να δει και τις πληροφορίες για την τάση – ρεύμα – συχνότητα – συντελεστή ισχύος του μήνα της επιλογής του. Ο μήνας που θα εμφανιστεί όταν η εφαρμογή τρέξει πρώτη φορά είναι ο τρέχων μήνας.

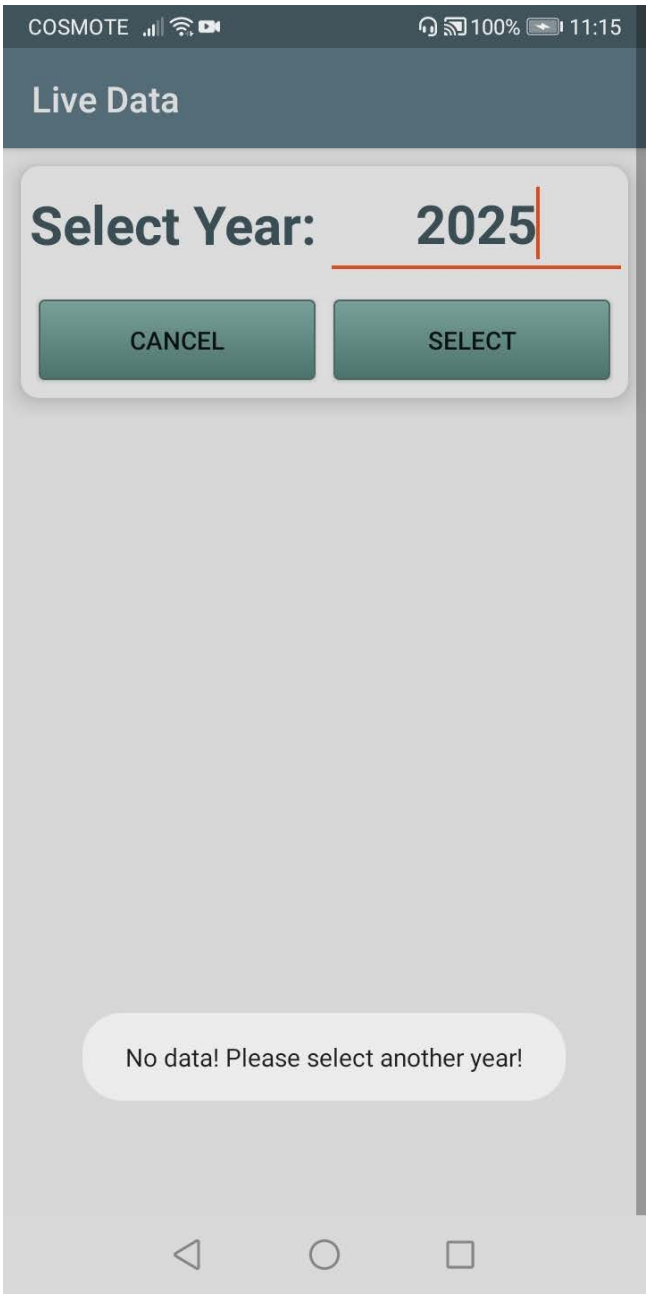
Μέσω της εφαρμογής μπορεί ο χρήστης να ενημερώνεται για το ποιός μήνας είναι ο πιο ενεργοβόρος και ποιά είναι η μέγιστη κατανάλωση ισχύος ώστε να μπορεί να εκτιμά και την αιτία της.

Η επιλογή του μήνα γίνεται από το αντίστοιχο μενού πατώντας πάνω στον μήνα ή / και το έτος. Σε περίπτωση που δεν επιλέξει κάποιο μήνα, αυτόματα επιλέγεται ο μήνας που είχε επιλεγεί τελευταία ενώ για το έτος, εφόσον ο χρήστης εισάγει κάποια έγκυρη τιμή, του δίνεται και η πληροφορία αν έχει ή όχι στοιχεία η βάση για το έτος της επιλογής του.



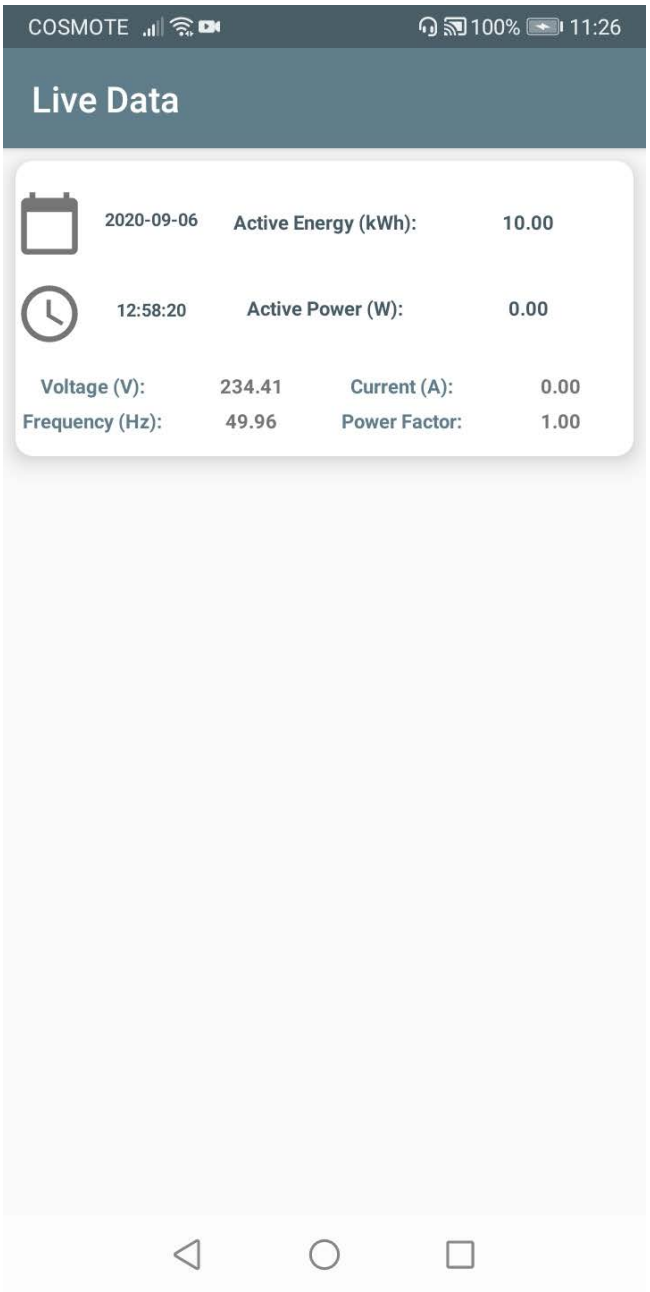


Στην περίπτωση που ο χρήστης θα ήθελε να διερευνήσει μια συγκεκριμένη μέρα προκειμένου να δει τα στοιχεία της και να ελέγξει τις μετρήσεις που έχουν καταχωρηθεί, μπορεί να επιλέγει το εικονίδιο λίστας προκειμένου να εισέλθει στην αντίστοιχη λειτουργία. Να σημειώσουμε ότι η λειτουργία αυτή επιλέχθηκε να δείχνει τα στοιχεία ανά ημέρα του μήνα και όχι ανά μήνα ή έτος ή όποια άλλη κατηγορία, για λόγους ταχύτητας απόκρισης της εφαρμογής. Κάθε μέρα, το πολύ, δίνονται 144 στοιχεία μια που η βάση ανανεώνεται κάθε 10 λεπτά. Αυτός είναι ένας αποδεκτός αριθμός στοιχείων τα οποία μπορούν άμεσα να φανούν στην οθόνη του χρήστη χωρίς αναμονή.

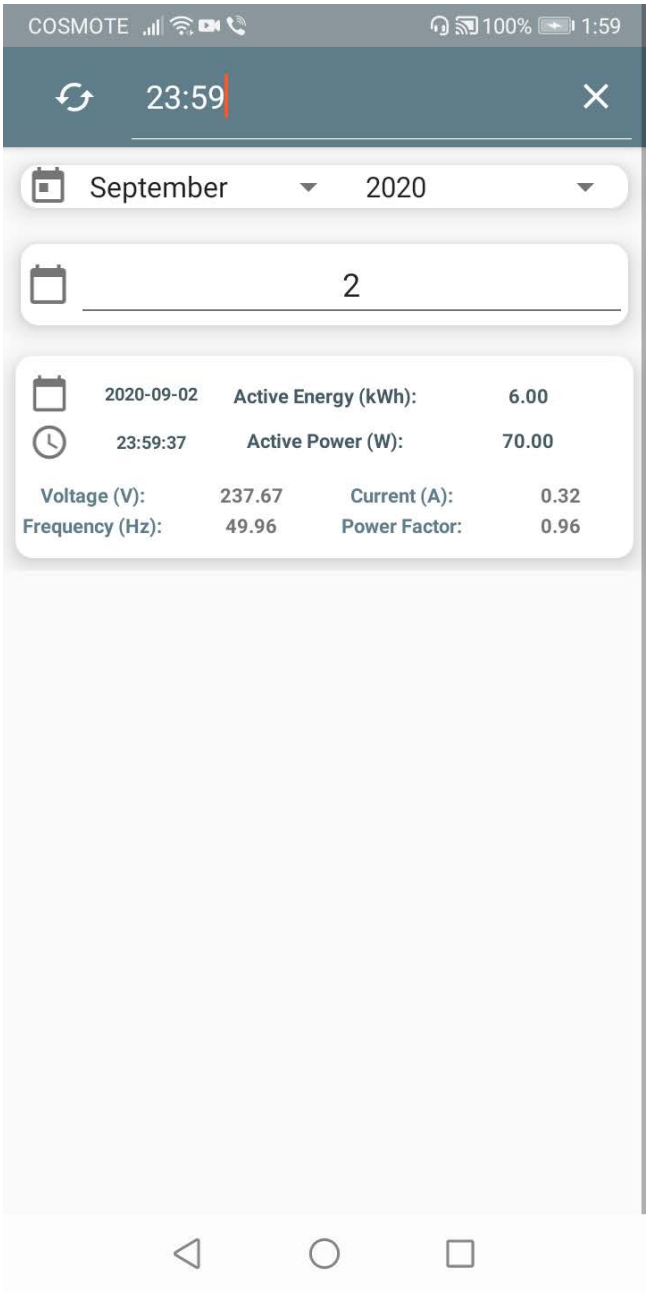


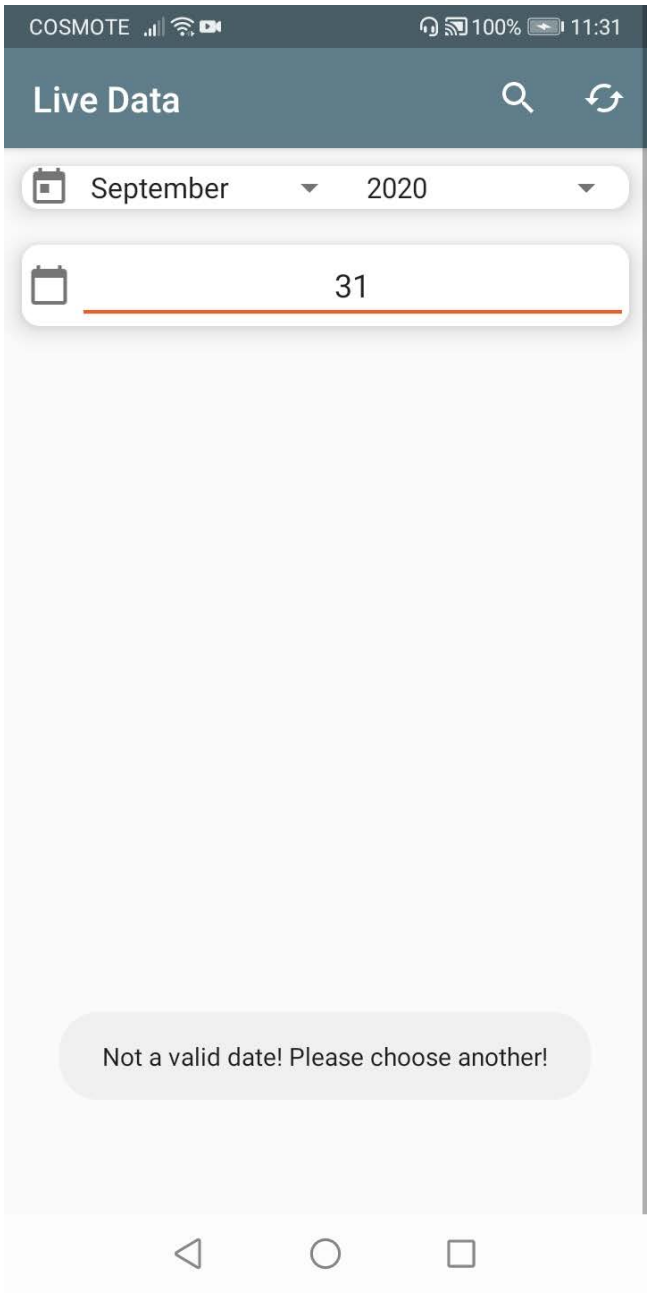
Επιπλέον, πατώντας πάνω σε οποιαδήποτε καταχώρηση όπως φαίνεται στη λίστα που εμφανίζεται ανά μέρα του μήνα ενός συγκεκριμένου έτους, μπορεί να απομονώσει τα στοιχεία αυτά ο χρήστης σε μια ξεχωριστή οθόνη. Η λίστα αυτή μπορεί να φιλτραριστεί ανάλογα με το κείμενο ή λέξη / κλειδί που θα εισάγει ο χρήστης στο πεδίο της αναζήτησης όπως αναδεικνύεται πατώντας πάνω στο κουμπάκι του μεγεθυντικού φακού.

Σε περίπτωση που ο χρήστης εισάγει μια λάθος ημερομηνία π.χ. 31.09.2020, τότε η εφαρμογή θα τον ενημερώσει άμεσα ότι αυτή του η επιλογή ήταν λανθασμένη με το αντίστοιχο μήνυμα.



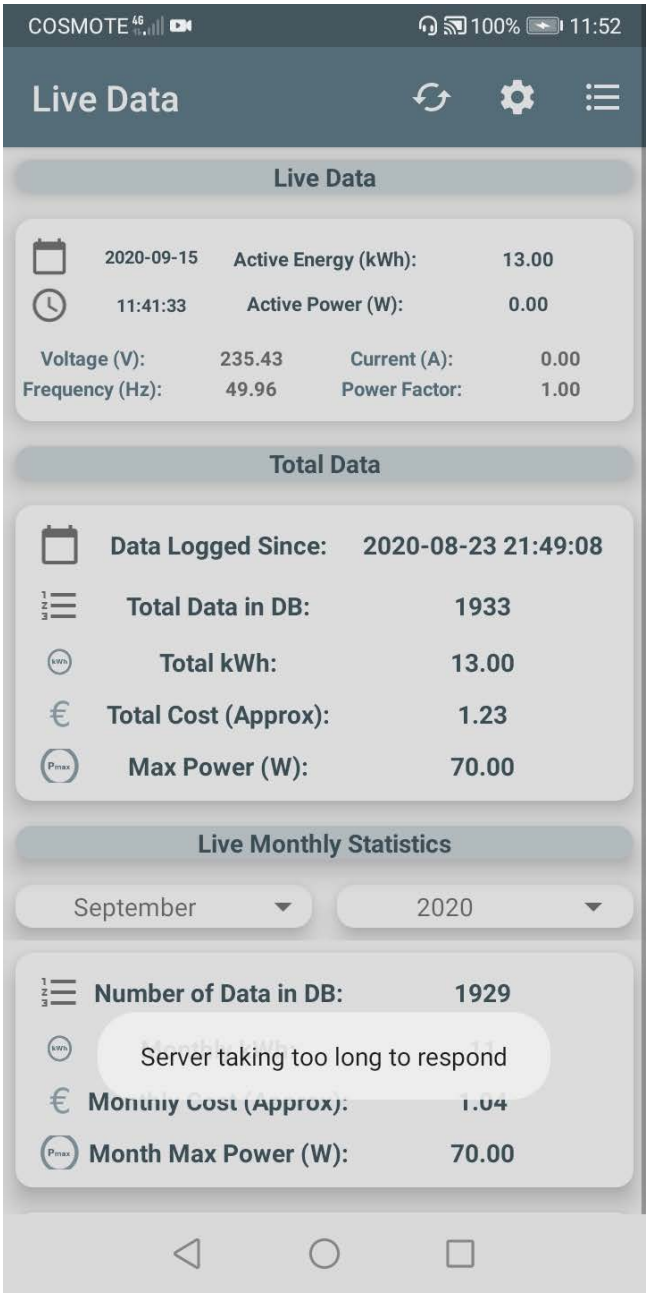
Όπως αναφέραμε και από την αρχή, η εφαρμογή αυτή υποστηρίζει έναν ενεργειακό μετρητή. Δεν μπορεί να υποστηρίξει στην παρούσα μορφή παραπάνω. Κι αυτό γιατί στη βάση θεωρείται ότι τα στοιχεία ανά χρήστη προκύπτουν από μια διάταξη. Επιπλέον, αν και δοκιμάστηκε για σχεδόν 2 μήνες, αυτό δεν σημαίνει ότι δεν υπάρχουν σημεία που πιθανώς να μην έχουν διερευνηθεί και να χρειάζονται τροποποίησης. Επιπλέον, θα ήταν χρήσιμο να δοκιμαστεί σε ευρύτερο πλαίσιο ώστε οι γνώμες διάφορων τύπων χρηστών να μπορούν να ληφθούν υπόψη και πιθανώς να ενσωματωθούν σε κάποια μελλοντική έκδοση της εφαρμογής. Προς το παρόν, όμως, η εφαρμογή όπως λειτουργεί μέχρι σήμερα, όχι μόνο δίνει άμεση πληροφορία για τις τιμές του ενεργειακού μετρητή αλλά μπορεί και να υποστηρίξει μια σχετικά λεπτομερή καταγραφή στοιχείων κατανάλωσης και δικτύου.


COSMOTE
100% 1:59
23:59
September
2020
2
2020-09-02
Active Energy (kWh):
6.00
23:59:37
Active Power (W):
70.00
Voltage (V):
237.67
Current (A):
0.32
Frequency (Hz):
49.96
Power Factor:
0.96


COSMOTE
100% 11:31
Live Data
September
2020
31
Not a valid date! Please choose another!

Επίλογος

Αναπτύχθηκε ένα ολοκληρωμένο σύστημα καταγραφής και ελέγχου κατανάλωσης για μια οικιακή γραμμή φωτισμού με έναν ρευματοδότη. Το σύστημα υποστηρίζει πολλούς διαφορετικούς χρήστες καθένας εκ των οποίων μπορεί να διαχειρίζεται στοιχεία από ένα στοιχείο καταγραφής. Στην περίπτωσή μας το στοιχείο αυτό ήταν ένας ενεργειακός μετρητής. Το σύστημα όχι μόνο παρέχει πληροφορίες για την κατανάλωση αλλά μπορεί και να δώσει στοιχεία για το δίκτυο όπως π.χ. πιθανές υπερ-τάσεις ή αυξομειώσεις στην συχνότητα. Παράλληλα δίνει πληροφορίες για το είδος των συσκευών που συνδέονται στη γραμμή που παρακολουθεί εντοπίζοντας πιθανά σφάλματα ή παρατυπίες. Για παράδειγμα, στις 02.09.2020 το ρεύμα της γραμμής ήταν 0.32 A, η τάση 237.67 V και ο συντελεστής ισχύος 0.96. Το γινόμενο αυτών των 3 μεγεθών θα πρέπει να δώσει κάτι κοντά στα 70 W. Αν αυτό είναι λάθος, θα πρέπει να αναθεωρηθεί το κύκλωμα.



Παρατηρούμε ότι το γινόμενο δίνει μια τιμή στα 73 W. Επειδή, όμως, ο ενεργειακός μετρητής έχει ακρίβεια 0.01 kW δηλαδή 10 W σύμφωνα με τον πίνακα της σελίδας 8, δεν μπορεί να μετρήσει 73 W αλλά μετρά 70 W. Αυτό είναι φυσιολογικό και θεωρούμε ότι το κύκλωμα λειτουργεί σωστά. Η δικιά μας εφαρμογή αν και έχει ουσιαστικά οφέλη για τον εκάστοτε χρήστη έχει και αρκετά μειονεκτήματα τα οποία θα αναλύσουμε λίγο εδώ.

Το βασικό της μειονέκτημα είναι ότι εξαρτάται από το δίκτυο wifi. Εφόσον το δίκτυο δεν είναι διαθέσιμο σταματά να λειτουργεί.

Επίσης, προκειμένου το σύστημα να εγκατασταθεί σε οποιοδήποτε σπίτι ή χώρο θα πρέπει να έχει, πέρα από τον χώρο στον πίνακα για την τοποθέτηση του ενεργειακού μετρητή, και χώρο ώστε να τοποθετηθεί το arduino που καλό θα ήταν να μην κρέμεται εκτός του πίνακα. Το arduino, από την άλλη μεριά χρειάζεται τη δικιά του τροφοδοσία κάτι που καθιστά το σύστημα λίγο δύσκολο να εγκατασταθεί εκ των υστέρων σε κάποια εγκατάσταση. Θα πρέπει να έχει προβλεφθεί η χρήση ενός συστήματος modbus σε κάποιο μέρος ώστε να μπορέσει να τοποθετηθεί το σύστημά μας με ασφάλεια.

Επίσης, η εφαρμογή δεν υποστηρίζει παραπάνω από έναν ενεργειακούς μετρητές όπως αναφέραμε αρκετές φορές ήδη στην παρούσα εργασία. Επιπλέον, το κόστος αγοράς των εξαρτημάτων καθώς και αυτό της εγκατάστασης ίσως να είναι απαγορευτικά για τον μέσο χρήστη. Στο μέλλον, πιθανώς, να δίνεται η δυνατότητα μιας πιο οικονομικής και ασφαλούς επιλογής.

Μελλοντικές επεκτάσεις

Η παρούσα εργασία είναι προπομπός πολλών άλλων που έχουν στόχο την ενημέρωση του χρήστη από απόσταση. Επιπλέον, θα ήταν ιδιαίτερα χρήσιμο, πέρα από την ενημέρωση, ο χρήστης να μπορεί από απόσταση να επέμβει στο σύστημά του και να ενεργοποιήσει / απενεργοποιήσει συσκευές ή ολόκληρες γραμμές. Συσκευές όπως π.χ. ανιχνευτές φωτιάς ή καπνού θα μπορούσαν να ελέγχονται από μακρυά και ο χρήστης να ενημερώνεται άμεσα για την κατάστασή τους. Στην αγορά κυκλοφορούν επίσης και ειδικές διατάξεις όπου ο χρήστης μπορεί από απόσταση να ενεργοποιήσει ή να απενεργοποιήσει κεντρικά ρελέ ή διακόπτες, δίνοντάς του μεγαλύτερη ευελιξία και ευκολία. Μια εφαρμογή android η οποία θα έχει π.χ. επικαιροποιημένη εικόνα της κατάστασης του πίνακα του χρήστη ανά πάσα στιγμή με δυνατότητα περαιτέρω ελέγχου από αυτόν, θα ήταν, ίσως, μια καλή επόμενη προσπάθεια. Με την εργασία αυτή όχι μόνο εντρυφήσαμε στα άδυτα της τεχνολογίας modbus αλλά είδαμε και πως μπορεί αυτή να ενσωματωθεί σε διαφορετικές εφαρμογές που τρέχουν σε μη προφανείς πλατφόρμες όπως π.χ. το arduino, ώστε να δώσει ένα χρηστικό αποτέλεσμα που θα έχει όφελος για τον τελικό χρήστη.

Πηγές

1. https://gs.statcounter.com/os-market-share/mobile/worldwide

2. https://www.udemy.com/course/mastering-modbus-rs485-network-communication/learn/lecture/405382#overview
3. https://www.udemy.com/course/mastering-modbus-tcpip-network-communication/learn/lecture/21223114#overview
4. https://www.udemy.com/course/learn-android-application-development-y/learn/lecture/11187982#overview
5. https://www.arduino.cc/
6. https://create.arduino.cc/projecthub/123325/monitor-your-energy-bill-via-modbus-mkr-wifi-1010-and-rs485-814e5e?ref=part&ref_id=77155&offset=0
7. https://create.arduino.cc/projecthub
8. https://www.arduino.cc/reference/en/
9. www.androidhive.info/
10. https://developer.android.com/
11. https://developer.android.com/docs
12. https://developer.android.com/reference
13. https://www.hager.gr/
14. https://www.hager.gr/e-catalogue/6049.htm
15. https://carbon.now.sh/

Παράρτημα

Κώδικας Arduino

```
/*
 *  This sketch connects the arduino MKR Wifi 1010 board to a wireless network
 *  and sends the Modbus data read by the MKR 485 Shield to a server.
 *
 *  MKR 485 Shield Configuration:
 *
 *  Half Duplex (no need for bidirectional communication, i.e. Full Duplex)
 *  ISO GND connected to GND of the Modbus RTU sensor (Hager ECR140D Energy Meter)
 *  - Y connected to (+) of the Modbus RTU sensor (Hager ECR140D Energy Meter)
 *  - Z connected to (-) of the Modbus RTU sensor (Hager ECR140D Energy Meter)
 *  - Jumper positions
 *    - FULL set to OFF
 *    - Z \/ Y set to ON
 *
 *   created by Eleftheria D.M. Kavoussanaki
 */

#include <ArduinoModbus.h>
#include <SPI.h>
#include <WiFiNINA.h>

#include "arduino_secrets.h"


char ssid[] = SECRET_SSID;    // network SSID (name)
char pass[] = SECRET_PASS;    // network password (use for WPA, or use as key for WEP)
String username = SECRET_USERNAME;  // server account username
String password = SECRET_PASSWORD; // server account password
String string = "";
```

```
int status = WL_IDLE_STATUS;

char server[] = SERVER_ADDRESS;    // name address for the server (with DNS)

/*
 *  Initialize the Ethernet client library
 *  with the IP address and port of the server
 *  that you want to connect to (port 80 is default for HTTP):
 */

WiFiClient client;

void setup() {

  /*
   *  Initialize serial and wait for port to open:
   */
  Serial.begin(9600);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  /*
  *   check if firmware needs to be updated (not essential)
  */
  /*String fv = WiFi.firmwareVersion();
```

```
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }*/

  /*
   *    Attempt to open modbus communication @ 9600 baud rate
   *    Configure Modbus RTU sensor (Hager ECR140D Energy Meter)
   *    to send data at this baud rate (default for Hager ECR140D 19200 baud rate)
   *
  */

  if (!ModbusRTUClient.begin(9600)) {
    Serial.println("Failed to start Modbus RTU Client!");
    //do not continue if failed
    while (1);
  }

  delay(10000);

}

void loop() {

  /*
   *
   * Read Modbus data
   *
   */

   float voltage = readVoltage();
   string = "Voltage: ";
   string += voltage;
   Serial.println(string);
   delay(100);
```

```
float current = readCurrent();
string = "Current: ";
string += current;
Serial.println(string);
 delay(100);
float frequency = readFrequency();
string = "Frequency: ";
string += frequency;
Serial.println(string);

 delay(100);
float power_factor = readPF_IEC();
string = "PF: ";
string += power_factor;
Serial.println(string);
 delay(100);
float active_power = readActive_power();
string = "Active Power: ";
string += active_power;
Serial.println(string);
 delay(100);
float reactive_power = readReactive_power();
string = "Reactive Power: ";
string += reactive_power;
Serial.println(string);
 delay(100);
float apparent_power = readApparent_power();
 string = "Apparent Power: ";
string += apparent_power;
Serial.println(string);
 delay(100);
float active_energy_in = readActiveEnergyIn();
string = "Active Energy In: ";
string += active_energy_in;
```

```
Serial.println(string);
 delay(100);
float active_energy_out = readActiveEnergyOut();
string = "Active Energy Out: ";
string += active_energy_out;
Serial.println(string);
 delay(100);

 /*
  *
  * Read other variables via modbus
  *
  */

float active_power_l1 = readActive_power_L1();
string = "Active Power L1: ";
string += active_power_l1;
Serial.println(string);
 delay(100);

 float reactive_power_l1 = readReactive_power_L1();
string = "Reactive Power L1: ";
string += reactive_power_l1;
Serial.println(string);
 delay(100);

 float apparent_power_l1 = readApparent_power_L1();
string = "Apparent Power L1: ";
string += apparent_power_l1;
Serial.println(string);
 delay(100);

 float active_tariff = readActiveTariffNumber();
 string = "Active Tariff: ";
```

```
 string += active_tariff;
 Serial.println(string);
 delay(100);

 float active_energy_in_t1 = readActiveEnergyIn_T1();
 string = "Active Energy In T1: ";
 string += active_energy_in_t1;
 Serial.println(string);
 delay(100);

  float active_energy_out_t1 = readActiveEnergyOut_T1();
 string = "Active Energy Out T1: ";
 string += active_energy_out_t1;
 Serial.println(string);
 delay(100);

 float reactive_energy_in_t1 = readReactiveEnergyIn_T1();
 string = "Reactive Energy In T1: ";
 string += reactive_energy_in_t1;
 Serial.println(string);
 delay(100);

 float reactive_energy_out_t1 = readReactiveEnergyOut_T1();
 string = "Reactive Energy Out T1: ";
 string += reactive_energy_out_t1;
 Serial.println(string);
 delay(100);
/*
 *  Attempt to connect to Wifi network to send data
 */

 while (status != WL_CONNECTED) {
 Serial.print("\n\nAttempting to connect to SSID: ");
 Serial.println(ssid);
```

```
   // Connect to WPA/WPA2 network.
   status = WiFi.begin(ssid, pass);

   // wait 10 seconds for connection:
   delay(10000);
  }

 //Serial.println("Connected to wifi");
 //printWifiStatus();


 Serial.println("\nStarting connection to server...");
 // if you get a connection, report back via serial:
 if (client.connect(server, 80)) {
   Serial.println("Connected to server");
   delay(100);
   // Make a HTTP request:
   String myString = "GET /dissertation/senddata_arduino.php?email=" + username +
"&password=" + password + "&active_energy_in=" + active_energy_in +
"&active_energy_out=" + active_energy_out + "&active_power=" + active_power
+"&voltage="+ voltage
+"&current="+current+"&frequency="+frequency+"&power_factor="+power_factor+"
HTTP/1.1";
   client.println(myString);
   client.println("Host: moijoune.net");
   client.println("Connection: close");
   client.println();
 }
 // if there are incoming bytes available
 // from the server, read them and print them:
 while (client.available()) {
   char c = client.read();
   Serial.write(c);
 }
```

```
  //wait for 1 min until the next run

  delay(60000);

  // if the server's disconnected, stop the client:
  /*if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting from server.");
    client.stop();

    // do nothing forevermore:
    while (true);
  }*/
}


float readVoltage(){
  float volt = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB000: The HEX address of the Voltage Data (V(L1-N))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, voltage is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB000, 0x01)) { //make the call to the register
    Serial.print("Failed to read voltage! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
```

```
  uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
  /*
   *
   * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we need to divide by 100 to get the
     * actual value in Volt
   *
   */
  volt = register_data/100.0;
 }
 return volt;
}

float readFrequency(){
 float frequency = 0.;
 /*
  *
  * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
  * HOLDING_REGISTERS are the registers to be read
  * 0xB006: The HEX address of the Frequency Data (F))
  * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, frequency is kept in 1 register)
  *
  */

 if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB006, 0x01)) { //make the call to the register
  Serial.print("Failed to read frequency! ");
  Serial.println(ModbusRTUClient.lastError()); //error handler
 }else{
  uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
  /*
   *
```

```
    * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we need to divide by 100 to get the
    * actual value in Hz
    *
    */
   frequency = register_data/100.0;
  }
  return frequency;
}

float readPF_IEC(){
  float pf = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB017: The HEX address of the Power Factor Data (PF(ΣL) IEC (§))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, power factor is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB017, 0x01)) { //make the call to the register
    Serial.print("Failed to read power factor! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.001 (Hager ECR140D Energy Meter) we need to divide by 1000 to get the
     * actual value
     *
```

```
    */
    pf = register_data/1000.0;
  }
  return pf;
}

float readPF_IEEE(){
  float pf = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB018: The HEX address of the Power Factor Data (PF(ΣL) IEEE (§))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, power factor is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB018, 0x01)) { //make the call to the register
    Serial.print("Failed to read power factor! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.001 (Hager ECR140D Energy Meter) we need to divide by 1000 to get the
     * actual value
     *
     */
    pf = register_data/1000.0;
  }
  return pf;
```

```
}


float readPF_L1_IEC(){
  float pf = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB02B: The HEX address of the Power Factor Data (PF(L1) IEC (§))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, power factor is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB02B, 0x01)) { //make the call to the register
    Serial.print("Failed to read power factor! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.001 (Hager ECR140D Energy Meter) we need to divide by 1000 to get the
     * actual value
     *
     */
    pf = register_data/1000.0;
  }
  return pf;
}
```

```
float readPF_L1_IEEE(){
  float pf = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB02E: The HEX address of the Power Factor Data (PF(L1) IEEE (§))
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, power factor is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB02E, 0x01)) { //make the call to the register
    Serial.print("Failed to read power factor! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.001 (Hager ECR140D Energy Meter) we need to divide by 1000 to get the
     * actual value
     *
     */
    pf = register_data/1000.0;
  }
  return pf;
}



float readActiveTariffNumber(){
  float tariff = 0.;
```

```
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB0B1: The HEX address of the Active Tariff (Active Tariff)
   * 0x01: The HEX number of registers to be read (For Hager ECR140D Energy Meter, active tariff is kept in 1 register)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB0B1, 0x01)) { //make the call to the register
    Serial.print("Failed to read active tariff! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    uint16_t register_data = ModbusRTUClient.read();  //read data from the buffer (U16)
    /*
     *
     * Since the resolution is 0.001 (Hager ECR140D Energy Meter) we need to divide by 1000 to get the
     * actual value
     *
     */
    tariff = register_data;
  }
  return tariff;
}


/*
 *
 * Read Double Registers
 *
 */
```

```
float readCurrent(){
  float current = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB009: The HEX address of the Current Data (I(L1))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, current is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB009, 0x02)) { //make the call to the register
    Serial.print("Failed to read current! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    uint32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (U16)
    uint32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (U16)
    uint32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two U16 to form a U32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the current in mA. To get it in A, we need to divide by 1000.0.
     *
     */
    current = register_data/1000.0;
  }
  return current;
}
```

```
float readActive_power(){
  float a_power = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB019: The HEX address of the Active Power Data (P(ΣL))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, active power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB011, 0x02)) { //make the call to the register
    Serial.print("Failed to read power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kW when we divide by 100. To convert to W we need to multiply by 1000.
     *
     */
    a_power = (register_data/100.0)*1000.0;
  }
  return a_power;
}
```

```
float readReactive_power(){
  float q_power = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB01F: The HEX address of the Reactive Power Data (Q(ΣL))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB013, 0x02)) { //make the call to the register
    Serial.print("Failed to read reactive power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kVar when we divide by 100. To convert to Var we need to multiply by 1000.
     *
     */
    q_power = (register_data/100.0)*1000.0;
  }
  return q_power;
}
```

```
float readApparent_power(){
  float s_power = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB025: The HEX address of the Current Data (S(ΣL))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB015, 0x02)) { //make the call to the register
    Serial.print("Failed to read apparent power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kVA when we divide by 100. To convert to VA we need to multiply by 1000.
     *
     */
    s_power = (register_data/100.0)*1000.0;
  }
  return s_power;
}
```

```
float readActive_power_L1(){
  float powerl1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB019: The HEX address of the Active Power Data (P(L1))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, active power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB019, 0x02)) { //make the call to the register
    Serial.print("Failed to read power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println("Active Power L1: ");
    //Serial.println(register_data1);
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println(register_data2);
    //Serial.println();
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kW when we divide by 100. To convert to W we need to multiply by 1000.
     *
```

```
    */
    powerl1 = (register_data/100.0)*1000.0;
  }
  return powerl1;
}

float readReactive_power_L1(){
  float q_powerl1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB01F: The HEX address of the Reactive Power Data (Q(L1))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB01F, 0x02)) { //make the call to the register
    Serial.print("Failed to read reactive power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kVar when we divide by 100. To convert to Var we need to multiply by 1000.
     *
```

```
    */
    q_powerl1 = (register_data/100.0)*1000.0;
  }
  return q_powerl1;
}

float readApparent_power_L1(){
  float s_powerl1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB025: The HEX address of the Current Data (S(L1))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB025, 0x02)) { //make the call to the register
    Serial.print("Failed to read apparent power for L1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 0.01 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kVA when we divide by 100. To convert to VA we need to multiply by 1000.
     *
```

```
    */
    s_powerl1 = (register_data/100.0)*1000.0;
  }
  return s_powerl1;
}

float readActiveEnergyIn(){
  float active_energy_in = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB060: The HEX address of the Active Energy In Data (Ea+ (ΣT))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB060, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy in! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println("Active energy in: ");
    //Serial.println(register_data1);
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println(register_data2);
    //Serial.println();
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
```

```
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    active_energy_in = register_data;
  }
  return active_energy_in;
}

float readReactiveEnergyIn(){
  float reactive_energy_in = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB062: The HEX address of the Reactive Energy In Data (Er+ (ΣT))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB062, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy in! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
```

```
     * of the power in kWh.
     *
     */
    reactive_energy_in = register_data;
  }
  return reactive_energy_in;
}

float readActiveEnergyOut(){
  float active_energy_out = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB064: The HEX address of the Active Energy Out Data (Ea+ (ΣT))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB064, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy out! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println("Active energy out: ");
    //Serial.println(register_data1);
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println(register_data2);
    //Serial.println();
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
```

```
   /*
    *
    * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
    * of the power in kWh.
    *
    */
   active_energy_out = register_data;
  }
  return active_energy_out;
}


float readReactiveEnergyOut(){
  float reactive_energy_out = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB066: The HEX address of the Active Energy Out Data (Er- (ΣT))
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB066, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy out! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
```

```
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    reactive_energy_out = register_data;
  }
  return reactive_energy_out;
}

float readActiveEnergyIn_T1(){
  float active_energy_in_t1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB0C0: The HEX address of the Active Energy In for Tariff 1 Data (Ea+ [T1])
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB0C0, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy in for tariff 1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println("Active energy in for tariff 1: ");
    //Serial.println(register_data1);
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
    //Serial.println(register_data2);
```

```
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    Serial.println();
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    active_energy_in_t1 = register_data;
  }
  return active_energy_in_t1;
}

float readReactiveEnergyIn_T1(){
  float reactive_energy_in_t1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB0E0: The HEX address of the Reactive Energy In Data for Tariff 1 (Er+ [T1])
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB0E0, 0x02)) { //make the call to the register
    Serial.print("Failed to read reactive energy in for tariff 1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
```

```
   int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    reactive_energy_in_t1 = register_data;
  }
  return reactive_energy_in_t1;
}

float readActiveEnergyOut_T1(){
  float active_energy_out_t1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB0D0: The HEX address of the Active Energy Out Data for Tariff 1 (Ea- [T1])
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB0D0, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy out! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
```

```
   int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a S32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    active_energy_out_t1 = register_data;
  }
  return active_energy_out_t1;
}


float readReactiveEnergyOut_T1(){
  float reactive_energy_out_t1 = 0.;
  /*
   *
   * 0x01: Modbus slave ID (Hager ECR140D Energy Meter Addr = 1)
   * HOLDING_REGISTERS are the registers to be read
   * 0xB0F0: The HEX address of the Active Energy Out Data for Tariff 1(Er- [T1])
   * 0x02: The HEX number of registers to be read (For Hager ECR140D Energy Meter, reactive power is kept in 2 registers)
   *
   */

  if (!ModbusRTUClient.requestFrom(0x01, HOLDING_REGISTERS, 0xB0F0, 0x02)) { //make the call to the register
    Serial.print("Failed to read active energy out for tariff 1! ");
    Serial.println(ModbusRTUClient.lastError()); //error handler
  }else{
    //2 registers!
    int32_t register_data1 = ModbusRTUClient.read();  //read data from the buffer (S16)
    int32_t register_data2 = ModbusRTUClient.read();  //read data from the buffer (S16)
```

```
    int32_t register_data = (register_data1 << 16) | register_data2; // bitwise add two S16 to form a
S32
    /*
     *
     * Since the resolution is 1 (Hager ECR140D Energy Meter) we get the actual value
     * of the power in kWh.
     *
     */
    reactive_energy_out_t1 = register_data;
  }
  return reactive_energy_out_t1;
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

Κώδικας HTML

Index.php

```
<?php
session_start();
$id = session_id();
if(isset($_COOKIE['USER_EMAIL'])){
        $email = $_COOKIE['USER_EMAIL'];
}
else{
        $email = "";
}
?>

<html>
<head>
<title> :: Energy Data :: </title>
<style>
body {
   background: black;
}

.digits {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translateX(-50%) translateY(-50%);
   color: #17D4FE;
   font-size: 60px;
   font-family: Orbitron;
   letter-spacing: 7px;
}
</style>
```

```
<script src="md5.js"></script>
</head>
<body>
<div id="myData" class="digits" onload="showData()"></div>
<script>

function getEmail(){
	return "<?php echo $email; ?>";
}

function activateFunc(){
	var email = getEmail();
	email = email.replace(/\s/g,'');
	//alert("2. " + email);
	if(email !== ''){
		//alert("2. " + email);
		showData();
	}
	else {

		var text = "Please <a href='system_login.php'>login</a> to view data! Or <a href='register_user.php'>register</a>!";
		//alert("2.2 " + email + ". " + text);
		document.getElementById("myData").innerHTML = text;
		//document.getElementById("myData").textContent = text;
	}
}

function showData() {
	var email = getEmail();
	//alert("1. " + email);
	var date = "123";
	var active_power = 0;
	var voltage = 0;
```

```
var active_energy_in = 0;
var active_energy_out = 0;
var current = 0;
var frequency = 0;
var pf = 1;

var text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " kWh<br/>Active Energy Out: " + active_energy_out + " kWh";
if (email.length == 0) {
	//text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " J<br/>Active Energy Out: " + active_energy_out + " J";
	document.getElementById("myData").innerHTML = "Please <a href='system_login.php'>login</a> to view data! Or <a href='register_user.php'>register</a>";
	//document.getElementById("myData").textContent = text;
}
else {
	//document.getElementById("myData").innerHTML = email;
	text = "No Data yet to show!";
	var xmlhttp = new XMLHttpRequest();
	xmlhttp.onreadystatechange = function() {
		if (this.readyState == 4 && this.status == 200) {
			//document.getElementById("myData").innerHTML = email;
			var json = JSON.parse(this.responseText);
			if(Object.keys(json.values[0]).length !== 0){
				date = json.values[0].date;
				active_power = json.values[0].data.active_power;
				voltage = json.values[0].data.voltage;
				current = json.values[0].data.current;
				frequency = json.values[0].data.frequency;
				pf = json.values[0].data.power_factor;
```

```
                                active_energy_in = json.values[0].data.active_energy_in;
                                active_energy_out = json.values[0].data.active_energy_out;
                                text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " kWh<br/>Active Energy Out: " + active_energy_out + " kWh";
                                text += "<br/><br/><a href='user_logout.php'>Logout</a>";
                                document.getElementById("myData").innerHTML = text;
                        }
                        else if(json.value == 2){
                                document.getElementById("myData").innerHTML = "Please <a href='system_login.php'>login</a> first!";
                        }
                        else{
                                document.getElementById("myData").innerHTML = json.message;
                                //document.getElementById("myData").textContent = text;
                        }
                }
                else{
                        document.getElementById("myData").innerHTML = "Please wait!";
                        //document.getElementById("myData").textContent = "<br/><br/>Please wait!";
                }
        };
        xmlhttp.open("POST", "./readdata.php", true);
        var params = "&email=" + email;
        xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xmlhttp.send(params);
    }

    setTimeout(showData, 5000);
```

```
}

//activateFunc();
showData();
</script>

</body>
</html>
```

db_auth.php

```
<?php
	define('DB_USERNAME', ""); // db user
	define('DB_PASSWORD', ""); // db password (mention your db password here)
	define('DB_DATABASE', ""); // database name
	define('DB_SERVER', ""); // db server
	define('DB_PORT', "");
	//define('APP_KEY', ); // application key
?>
```

register.php

```
<?php
session_start();
require_once __DIR__ . '/db_auth.php';
$result = array();
//if(isset($_POST['app_key'])){
	if(isset($_POST['email']) && isset($_POST['password']) && isset($_POST['password_again'])){
		if($_POST['password'] == $_POST['password_again']){
			// Create connection
			$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_DATABASE, DB_PORT);
			// Check connection
```

```
if ($conn->connect_error) {
	$result["message"] = "Connection error: ".$conn->connect_error;
	$result["value"] = 0;
	echo json_encode($result);
	//die("Connection failed: " . $conn->connect_error);
}
else{
	$sql = "SELECT * FROM `userdata` WHERE `email` = '".$_POST['email']."' LIMIT 1";
	//echo $sql;
	$m_result = $conn->query($sql);
	//var_dump($m_result);
	if ($m_result->num_rows > 0) {
		$result["message"] = "This user has already registered! Reset password?";
		$result["value"] = 2;
		echo json_encode($result);
	}
	else{
		$sql = "INSERT INTO `userdata`(`email`, `password`) VALUES ('".$_POST['email']."','".md5($_POST['password'])."')";
		//echo $sql;
		// output data of each row
		$m_result = $conn->query($sql);
		//var_dump($m_result);
		if($conn->affected_rows == 1) {
			$result["message"] = "User has succefully registered! You can now login!";
			$result["value"] = 1;
			echo json_encode($result);
		}
		else{
			$result["message"] = "User has failed to register!";
			$result["value"] = 0;
```

```
                                        echo json_encode($result);
                                    }
                                }
                            }
                        }
                        else{
                            $result['message'] = "Passwords do not match!";
                            $result['value'] = 0;
                            echo json_encode($result);
                        }
                }
                else{
                        $result['message'] = "Essential info missing!";
                        $result['value'] = 0;
                        echo json_encode($result);
                }
/*}
else{
        $result['message'] = "Sorry! Not authorized to use this app!";
        $result['value'] = 0;
        echo json_encode($result);
}*/

?>
```

register_user.php

```
<?php
session_start();
?>

<html>
<head>
<title> :: Energy Data :: </title>
```

```
<style>
body {
   background: black;
}

.digits {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translateX(-50%) translateY(-50%);
   color: #17D4FE;
   font-size: 60px;
   font-family: Orbitron;
   letter-spacing: 7px;
}
</style>

<script>

function register() {
        var email = document.getElementById("email").value;
        var password1 = document.getElementById("password1").value;
        var password2 = document.getElementById("password2").value;

        if (email.length == 0 || password1.length == 0 || password2.length == 0) {
                //alert("Please provide an email/password! No blancs allowed!");
                //text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " J<br/>Active Energy Out: " + active_energy_out + " J";
                document.getElementById("result").innerHTML = "No email and/or password! <br/>Please <a href='register_user.php'>try again</a>!";
                //document.getElementById("myData").textContent = text;
        }
```

```
        else if(password1 !== password2){
                document.getElementById("result").innerHTML = "Passwords do not match! <br/>Please <a href='register_user.php'>try again</a>!";
        }
        else {
                var xmlhttp = new XMLHttpRequest();
                xmlhttp.onreadystatechange = function() {
                        if (this.readyState == 4 && this.status == 200) {
                                var json = JSON.parse(this.responseText);
                                //document.getElementById("result").innerHTML = json.value;
                                if(json.value == 1){
                                        document.getElementById("result").innerHTML = "Registration Successful! Ready to <a href='system_login.php'>login</a>!";
                                }
                                else if(json.value == 2){
                                        document.getElementById("result").innerHTML = "This user has already registered! <a href='reset_user.php'>Reset</a> password?";
                                }
                                else{
                                        document.getElementById("result").innerHTML = json.message;
                                }
                                //document.getElementById("myData").innerHTML = text;
                                //document.getElementById("myData").textContent = text;

                        }
                        else{
                                document.getElementById("result").innerHTML = "Please wait!";
                                //document.getElementById("myData").textContent = "<br/><br/>Please wait!";
                        }
                };
                xmlhttp.open("POST", "./register.php", true);
```

```
		var params = "&email=" + email + "&password=" + password1 + "&password_again=" + password2;
		xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
		xmlhttp.send(params);
	}

	//setTimeout(showData, 5000);

	return false;
}
</script>
</head>
<body>
<div id="result" class = "digits">
	<form id="myData" onsubmit="return register();">
	<table class="digits">
	<tr><td><label for="email">Email:</label></td><td>
	<input type="text" style="height:40px;" name="email" id = "email" size="40"></td></tr>
	<tr><td><label for="password1">Password:</label></td><td>
	<input type="password" style="height:40px;" name="password1" id = "password1" size="40"></td></tr>
	<tr><td><label for="password2">Password Again:</label></td><td>
	<input type="password" style="height:40px;" name="password2" id = "password2" size="40"></td></tr>
	<tr><td colspan="2" style="text-align: center;"><br/>
	<input type="submit" name="submit" value="Register" style="font-size: 25px;height:60px;width:220px"></td></tr>
	</table>
	</form>
</div>


</body>
```

```
</html>
?>
```

login.php

```
<?php
session_start();
$id = session_id();
require_once __DIR__ . '/db_auth.php';
$login_result = array();
//if(isset($_POST['app_key'])){
	if(isset($_POST['email']) && isset($_POST['password'])){
		if(!isset($_COOKIE['SESSION_ID']) || $_COOKIE['SESSION_ID'] != $id){
			// Create connection
			$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_DATABASE, DB_PORT);
			// Check connection
			if ($conn->connect_error) {
				$login_result['message'] = "Connection error: ".$conn->connect_error;
				$login_result['value'] = 0;
				echo json_encode($login_result);
				//die("Connection failed: " . $conn->connect_error);
			}
			else{
				$sql = "SELECT * FROM `userdata` WHERE `email` = '".$_POST['email']."' AND `password` = '".md5($_POST['password'])."' LIMIT 1";
				$result = $conn->query($sql);
				if ($result->num_rows > 0) {
					$login_result['message'] = "Logged in Successfully!";
					$login_result['value'] = 1;
					setcookie("SESSION_ID", $id, time() + (86400 * 30), "/");
					$row = $result->fetch_assoc();
```

```
                                        setcookie("USER_EMAIL", $_POST['email'], time() +
(86400 * 30), "/");
                                        $sql = "INSERT INTO `useractivity`(`email`, `session_id`,
`currentstatus`) VALUES ('".$_POST['email']."','".$id."','logged in')";
                                        $m_result = $conn->query($sql);
                                        //var_dump($m_result);
                                        if($conn->affected_rows != 1) {
                                                $login_result['value'] = 0;
                                        }
                                        echo json_encode($login_result);
                                }
                                else{
                                        $login_result['message'] = "Wrong username and / or
password! Reset password or Register?";
                                        $login_result['value'] = 2;
                                        echo json_encode($login_result);
                                }
                        }
                }
                else{
                        $login_result['message'] = "Already Logged In!";
                        $login_result['value'] = 2;
                        echo json_encode($login_result);
                }
        }
        else{
                $login_result['message'] = "Essential info missing!";
                $login_result['value'] = 0;
                echo json_encode($login_result);
        }
/*}
else{
        $login_result['message'] = "Sorry! Not authorized to use this app!";
        $login_result['value'] = 0;
```

```
        echo json_encode($login_result);
}*/

?>
```

system_login.php

```
<?php
session_start();
?>

<html>
<head>
<title> :: Energy Data :: </title>
<style>
body {
   background: black;
}

.digits {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translateX(-50%) translateY(-50%);
   color: #17D4FE;
   font-size: 60px;
   font-family: Orbitron;
   letter-spacing: 7px;
}
</style>

<script>
```

```
function login() {
	var email = document.getElementById("email").value;
	var password = document.getElementById("password").value;

	if (email.length == 0 || password.length == 0) {
		//alert("Please provide an email/password! No blancs allowed!");
		//text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " J<br/>Active Energy Out: " + active_energy_out + " J";
		document.getElementById("result").innerHTML = "No email and/or password! <br/>Please <a href='system_login.php'>try again</a>!";
		//document.getElementById("myData").textContent = text;
	}
	else {
		var xmlhttp = new XMLHttpRequest();
		xmlhttp.onreadystatechange = function() {
			if (this.readyState == 4 && this.status == 200) {
				var json = JSON.parse(this.responseText);
				if(json.value == 1){
					window.location.replace("http://localhost/data/");
				}
				else if(json.value == 2){
					document.getElementById("result").innerHTML = "Wrong username and / or password! <a href='reset_user.php'>Reset</a> password or <a href='register_user.php'>Register</a>?";
				}
				else{
					document.getElementById("result").innerHTML = json.message;
				}
				//document.getElementById("myData").innerHTML = text;
				//document.getElementById("myData").textContent = text;
```

```
                }
                else{
                        document.getElementById("result").innerHTML = "Please wait!";
                        //document.getElementById("myData").textContent = "<br/><br/>Please wait!";
                }
        };
        xmlhttp.open("POST", "./login.php", true);
        var params = "&email=" + email + "&password=" + password;
        xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xmlhttp.send(params);
    }

    //setTimeout(showData, 5000);

    return false;
}
</script>
</head>
<body>
<div id="result" class = "digits">
    <form id="myData" onsubmit="return login();">
    <table class="digits">
    <tr><td><label for="email">Email:</label></td><td>
    <input type="text" style="height:40px;" name="email" id = "email" size="40"></td></tr>
    <tr><td><label for="password">Password:</label></td><td>
    <input type="password" style="height:40px;" name="password" id = "password" size="40"></td></tr>
    <tr><td colspan="2" style="text-align: center;"><br/>
    <input type="submit" name="submit" value="Login" style="font-size: 25px;height:60px;width:220px"></td></tr>
    </table>
    </form>
```

```
</div>


</body>
</html>
```

reset.php

```
<?php
session_start();
$id = session_id();
require_once __DIR__ . '/db_auth.php';
$result = array();

if(isset($_POST['email']) && isset($_POST['password']) && isset($_POST['password_again'])){
	if (isset($_COOKIE['SESSION_ID'])) {
		unset($_COOKIE['SESSION_ID']);
		setcookie('SESSION_ID', null, -1, '/');
	}
	if (isset($_COOKIE['USER_EMAIL'])) {
		unset($_COOKIE['USER_EMAIL']);
		setcookie('USER_EMAIL', null, -1, '/');
	}
	if($_POST['password'] == $_POST['password_again']){
		// Create connection
		$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_DATABASE, DB_PORT);
		// Check connection
		if ($conn->connect_error) {
			$result['message'] = "Connection error: ".$conn->connect_error;
			$result['value'] = 0;
			echo json_encode($result);
			//die("Connection failed: " . $conn->connect_error);
```

```
            }
            else{
                $sql = "SELECT * FROM `userdata` WHERE `email` = '".$_POST['email']."' LIMIT 1";
                //echo $sql;
                $m_result = $conn->query($sql);
                //var_dump($m_result);
                if ($m_result->num_rows < 1) {
                    $result['message'] = "This user does not exist. Please register!";
                    $result['value'] =  2;
                    echo json_encode($result);
                }
                else{
                    $sql = "UPDATE `userdata` SET `password`= '".md5($_POST['password'])."' WHERE `email` = '".$_POST['email']."' LIMIT 1";
                    //echo $sql;
                    // output data of each row
                    $m_result = $conn->query($sql);
                    //var_dump($m_result);
                    if($conn->affected_rows == 1) {
                        $result["message"] = "Password reset successful! You can now login!";
                        $result["value"] =  1;
                        echo json_encode($result);
                    }
                    else{
                        $result["message"] = "Password was not reset!";
                        $result["value"] =  0;
                        echo json_encode($result);
                    }
                }
            }
        }
        else{
```

```
                $result['message'] = "Passwords do not match!";
                $result['value'] = 0;
                echo json_encode($result);
        }
}
else{
        $result['message'] = "Essential info missing!";
        $result['value'] = 0;
        echo json_encode($result);
}

?>
```

reset_user.php

```
<?php
session_start();
?>

<html>
<head>
<title> :: Energy Data :: </title>
<style>
body {
   background: black;
}

.digits {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translateX(-50%) translateY(-50%);
   color: #17D4FE;
```

```
    font-size: 60px;
    font-family: Orbitron;
    letter-spacing: 7px;
}
</style>

<script>

function reset_user() {
        //alert("OK");
        var email = document.getElementById("email").value;
        var password1 = document.getElementById("password1").value;
        var password2 = document.getElementById("password2").value;

        if (email.length == 0 || password1.length == 0 || password2.length == 0) {
                //alert("Please provide an email/password! No blancs allowed!");
                //text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " J<br/>Active Energy Out: " + active_energy_out + " J";
                document.getElementById("result").innerHTML = "No email and/or password! <br/>Please <a href='reset_user.php'>try again</a>!";
                //document.getElementById("myData").textContent = text;
        }
        else if(password1 !== password2){
                document.getElementById("result").innerHTML = "Passwords do not match! <br/>Please <a href='reset_user.php'>try again</a>!";
        }
        else {
                var xmlhttp = new XMLHttpRequest();
                xmlhttp.onreadystatechange = function() {
                        if (this.readyState == 4 && this.status == 200) {
                                var json = JSON.parse(this.responseText);
                                //document.getElementById("result").innerHTML = json.value;
```

```
                        if(json.value == 1){
                                document.getElementById("result").innerHTML =
"Password reset Successful! Ready to <a href='system_login.php'>login</a>!";
                        }
                        else if(json.value == 2){
                                document.getElementById("result").innerHTML = "This
user does not exist. Please <a href='register_user.php'>register</a>!";
                        }
                        else{
                                document.getElementById("result").innerHTML =
json.message;
                        }
                        //document.getElementById("myData").innerHTML = text;
                        //document.getElementById("myData").textContent = text;

                }
                else{
                        document.getElementById("result").innerHTML = "Please wait!";
                        //document.getElementById("myData").textContent =
"<br/><br/>Please wait!";
                }
        };
        xmlhttp.open("POST", "./reset.php", true);
        var params = "&email=" + email + "&password=" + password1 +
"&password_again=" + password2;
        xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
        xmlhttp.send(params);
    }

    //setTimeout(showData, 5000);

    return false;
}
```

```
</script>
</head>
<body>
<div id="result" class = "digits">
	<form id="myData" onsubmit="return reset_user();">
	<table class="digits">
	<tr><td><label for="email">Email:</label></td><td>
	<input type="text" style="height:40px;" name="email" id = "email" size="40"></td></tr>
	<tr><td><label for="password1">Password:</label></td><td>
	<input type="password" style="height:40px;" name="password1" id = "password1" size="40"></td></tr>
	<tr><td><label for="password2">Password Again:</label></td><td>
	<input type="password" style="height:40px;" name="password2" id = "password2" size="40"></td></tr>
	<tr><td colspan="2" style="text-align: center;"><br/>
	<input type="submit" name="submit" value="Reset Password" style="font-size: 25px;height:60px;width:220px"></td></tr>
	</table>
	</form>
</div>


</body>
</html>
?>
```

readdata.php

```
<?php
session_start();
$id = session_id();
require_once __DIR__ . '/db_auth.php';
$result = array();
```

```
//$test_data = $data = array("active_energy_in"=>12, "active_energy_out"=>13, "active_power"=>14, "voltage"=>15, "current"=>16, "frequency"=>17, "power_factor"=>18);
//if(isset($_POST['app_key'])){
	if(isset($_POST['email'])){
		if(isset($_COOKIE['SESSION_ID']) && $_COOKIE['SESSION_ID'] == $id && isset($_COOKIE['USER_EMAIL']) && $_POST['email'] == $_COOKIE['USER_EMAIL']){
			// Create connection
			$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_DATABASE, DB_PORT);
			// Check connection
			if ($conn->connect_error) {
				$result['message'] = "Connection error: ".$conn->connect_error;
				$result['value'] = 0;
				$result['values'][0] = array();
				echo json_encode($result);
				//die("Connection failed: " . $conn->connect_error);
			}
			else{
				$sql = "SELECT * FROM `rawdata` WHERE `email` = '".$_POST['email']."' ORDER BY `update_date` DESC LIMIT 1";
				$sql_res = $conn->query($sql);

				$result['message'] = "Data Read Successfully!";
				$result['value'] = 1;
				$result['values'][0] = array();
				//$result['values'][0] = array("date"=>"2","data"=>$test_data);
				while(($row = $sql_res->fetch_assoc())){
					$date = $row['update_date'];
					$data = array("active_energy_in"=>$row['active_energy_in'], "active_energy_out"=>$row['active_energy_out'], "active_power"=>$row['active_power'], "voltage"=>$row['voltage'], "current"=>$row['current'], "frequency"=>$row['frequency'], "power_factor"=>$row['power_factor']);
					$result['values'][0] = array("date"=> $date, "data"=>$data);
```

```
                        }
                        if(empty($result['values'])) {
                                $result['message'] = "No Data!";
                        }
                        echo json_encode($result);
                }
        }
        else{
                $result['message'] = "Please login first!";
                $result['value'] = 2;
                $result['values'][0] = array();
                echo json_encode($result);
        }
}
else{
        $result['message'] = "Essential Info missing!";
        $result['value'] = 0;
        $result['values'][0] = array();
        echo json_encode($result);
}
/*}
else{
        $result['message'] = "Sorry! Not authorized to use this app!";
        $result['value'] = 0;
        echo json_encode($result);
}*/



?>
```

logout.php

```
<?php
session_start();
$id = session_id();
require_once __DIR__ . '/db_auth.php';
$result = array();
//if(isset($_POST['app_key'])){
	if(isset($_COOKIE['USER_EMAIL'])){
		if(isset($_COOKIE['SESSION_ID']) && $_COOKIE['SESSION_ID'] == $id){
			// Create connection
			$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_DATABASE, DB_PORT);
			// Check connection
			if ($conn->connect_error) {
				$result['message'] = "Connection error: ".$conn->connect_error;
				$result['value'] = 0;
				echo json_encode($result);
				//die("Connection failed: " . $conn->connect_error);
			}
			else{
				$sql = "SELECT * FROM `useractivity` WHERE `email` = '".$_COOKIE['USER_EMAIL']."' AND `session_id` = '".$id."' LIMIT 1";
				$sql_result = $conn->query($sql);
				if ($sql_result->num_rows > 0) {
					$result['message'] = "Logged out Successfully!";
					$result['value'] = 1;
					$sql = "INSERT INTO `useractivity`(`email`, `session_id`, `currentstatus`) VALUES ('".$_COOKIE['USER_EMAIL']."','".$id."','logged out')";
					$m_result = $conn->query($sql);
					//var_dump($m_result);
					if($conn->affected_rows != 1) {
						$result['value'] = 0;
					}

					if (isset($_COOKIE['SESSION_ID'])) {
```

```
                unset($_COOKIE['SESSION_ID']);
                setcookie('SESSION_ID', null, -1, '/');
        }
        if (isset($_COOKIE['USER_EMAIL'])) {
                unset($_COOKIE['USER_EMAIL']);
                setcookie('USER_EMAIL', null, -1, '/');
        }
        if(!isset($_COOKIE['SESSION_ID'])                    &&
!isset($_COOKIE['USER_EMAIL'])){
                $result['message'] = "Logout successful!";
                $result['value'] = 1;
                //echo json_encode($result);
        }
        else{
                $result['message'] = "Logout unsuccessful!";
                $result['value'] = 0;
                //echo json_encode($result);
        }

        $_SESSION = array();


        // If it's desired to kill the session, also delete the session
cookie.
        // Note: This will destroy the session, and not just the
session data!
        if (ini_get("session.use_cookies")) {
                $params = session_get_cookie_params();
                setcookie(session_name(), '', time() - 42000,
                        $params["path"], $params["domain"],
                        $params["secure"], $params["httponly"]
                );
        }
        session_destroy();
```

```
                                        echo json_encode($result);
                                }
                                else{
                                        $result['message'] = "Wrong username and / or session? Is the user already logged out?";
                                        $result['value'] = 2;
                                        echo json_encode($result);
                                }
                        }
                }
                else{
                        $result['message'] = "Already Logged Out!";
                        $result['value'] = 2;
                        echo json_encode($result);
                }
        }
        else{
                $result['message'] = "Essential info missing!";
                $result['value'] = 0;
                echo json_encode($result);
        }
/*}
else{
        $login_result['message'] = "Sorry! Not authorized to use this app!";
        $login_result['value'] = 0;
        echo json_encode($login_result);
}*/

?>
```

user_logout.php

```
<?php
session_start();
$id = session_id();
if(isset($_COOKIE['USER_EMAIL'])){
        $email = $_COOKIE['USER_EMAIL'];
}
else{
        $email = "";
}
?>

<html>
<head>
<title> :: Energy Data :: </title>
<style>
body {
   background: black;
}

.digits {
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translateX(-50%) translateY(-50%);
   color: #17D4FE;
   font-size: 60px;
   font-family: Orbitron;
   letter-spacing: 7px;
}
</style>
</head>
```

```
<body>
<div id="result" class = "digits"></div>
<script>

function getEmail(){
	return "<?php echo $email; ?>";
}

function logout() {
	var email = getEmail();

	if (email.length == 0) {
		//alert("Please provide an email/password! No blancs allowed!");
		//text = "Date: " + date + "<br/>Active Power: " + active_power + " W<br/>Voltage: " + voltage + " V<br/>Current: " + current + " A<br/>Frequency: " + frequency + " Hz<br/>Power Factor: " + pf + "<br/>Active Energy In: " + active_energy_in + " J<br/>Active Energy Out: " + active_energy_out + " J";
		document.getElementById("result").innerHTML = "User unknown!";
		//document.getElementById("myData").textContent = text;
	}
	else {
		var xmlhttp = new XMLHttpRequest();
		xmlhttp.onreadystatechange = function() {
			if (this.readyState == 4 && this.status == 200) {
				var json = JSON.parse(this.responseText);
				if(json.value == 1){
					//window.location.replace("http://localhost/data/");
					document.getElementById("result").innerHTML = json.message;
				}
				else{
					document.getElementById("result").innerHTML = json.message;
				}
```

```
                        //document.getElementById("myData").innerHTML = text;
                        //document.getElementById("myData").textContent = text;

                }
                else{
                        document.getElementById("result").innerHTML = "Please wait!";
                        //document.getElementById("myData").textContent =
"<br/><br/>Please wait!";
                }
        };
        xmlhttp.open("POST", "./logout.php", true);
        var params = "&email=" + email;
        xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
        xmlhttp.send(params);
    }

    //setTimeout(showData, 5000);

    return false;
}
logout();

setTimeout(function () {
    window.location.replace("http://localhost/data/"); //will redirect to your blog page (an ex:
blog.html)
}, 1000);
</script>
</body>
</html>
```

Κώδικας Android

MainActivity.java

```
package com.example.ptyxiakh2;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.core.view.MenuItemCompat;

import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.DataSetObserver;
import android.database.SQLException;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.SearchView;
import android.widget.Spinner;
import android.widget.SpinnerAdapter;
import android.widget.TextView;
import android.widget.Toast;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
```

```
import org.json.JSONException;
import org.json.JSONObject;
import org.w3c.dom.Text;

import java.net.URL;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    ActionBar actionBar;

    static int SELECT_MONTH_ACTIVITY = 1;
    static int SELECT_YEAR_ACTIVITY = 2;
    static int SETTINGS_ACTIVITY = 3;
    //static int DATETIME_PICKER_ACTIVITY = 4;

    TextView tvUpdateDate, tvActiveEnergyIn, tvUpdateTime, tvActivePower, tvVoltage,
tvCurrent, tvFrequency, tvPF;

    TextView tvStartDate, tvTotalDBData, tvTotalKWh, tvTotalCost, tvMaxPower;

    CardView monthCard, yearCard;

    TextView tvMonth, tvYear;

    TextView tvMonthlyDBData, tvMonthlyKWh, tvMonthlyCost, tvMonthlyMaxPower;

    TextView tvVmax, tvVmin, tvImax, tvImin, tvfMax, tvfMin, tvPfMax, tvPfMin;

    ArrayList<String> months = new ArrayList<>();
```

```
int currentMonth;

int currentYear;
int year_zero = 2020;

SharedPreferences prefs;
public static final String MY_PREF_FILENAME = "com.example.ptyxiakh2.userdata";

String kWh_cost;

String username, password;

String today, now;

String url1 = "http://www.moijoune.net/dissertation/readdata_android.php";
String url2 = "http://www.moijoune.net/dissertation/readalldata_android1.php";

AsyncTask getData = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    prefs = getSharedPreferences(MY_PREF_FILENAME, MODE_PRIVATE);

    kWh_cost = "0.09460";

    Date date = new Date();

    DateFormat dateFormat = new SimpleDateFormat("MM");
    currentMonth = Integer.parseInt(dateFormat.format(date).toString()) - 1;

    DateFormat dateFormat1 = new SimpleDateFormat("yyyy");
    currentYear = Integer.parseInt(dateFormat1.format(date).toString());

    DateFormat dateFormat2 = new SimpleDateFormat("yyyy-MM-dd");
```

```
today                   =                   dateFormat2.format(date).toString();

DateFormat      dateFormat3      =      new      SimpleDateFormat("HH:mm:ss");
now                     =                   dateFormat3.format(date).toString();

months.add("January");
months.add("February");
months.add("March");
months.add("April");
months.add("May");
months.add("June");
months.add("July");
months.add("August");
months.add("September");
months.add("October");
months.add("November");
months.add("December");

tvUpdateDate        =        (TextView)        findViewById(R.id.tvUpdateDate);
tvActiveEnergyIn     =     (TextView)     findViewById(R.id.tvActiveEnergyIn);
tvUpdateTime        =        (TextView)        findViewById(R.id.tvUpdateTime);
tvActivePower       =       (TextView)       findViewById(R.id.tvActivePower);
tvVoltage          =          (TextView)          findViewById(R.id.tvVoltage);
tvCurrent          =          (TextView)          findViewById(R.id.tvCurrent);
tvFrequency         =         (TextView)         findViewById(R.id.tvFrequency);
tvPF             =             (TextView)             findViewById(R.id.tvPF);

tvUpdateDate.setText(prefs.getString("tvUpdateDate","N/A"));
tvActiveEnergyIn.setText(prefs.getString("tvActiveEnergyIn",          "N/A"));
tvUpdateTime.setText(prefs.getString("tvUpdateTime",                  "N/A"));
tvActivePower.setText(prefs.getString("tvActivePower",                "N/A"));
tvVoltage.setText(prefs.getString("tvVoltage",                        "N/A"));
tvCurrent.setText(prefs.getString("tvCurrent",                        "N/A"));
tvFrequency.setText(prefs.getString("tvFrequency",                    "N/A"));
tvPF.setText(prefs.getString("tvPF","N/A"));

tvStartDate         =         (TextView)         findViewById(R.id.tvStartDate);
tvTotalDBData       =       (TextView)       findViewById(R.id.tvTotalDBData);
tvTotalKWh          =          (TextView)          findViewById(R.id.tvTotalKWh);
```

```
        tvTotalCost = (TextView) findViewById(R.id.tvTotalCost);
        tvMaxPower = (TextView) findViewById(R.id.tvMaxPower);

        tvStartDate.setText(prefs.getString("tvStartDate", "N/A"));
        tvTotalDBData.setText(prefs.getString("tvTotalDBData", "N/A"));
        tvTotalKWh.setText(prefs.getString("tvTotalKWh", "N/A"));
        tvTotalCost.setText(prefs.getString("tvTotalCost", "N/A"));
        tvMaxPower.setText(prefs.getString("tvMaxPower", "N/A"));

        monthCard = (CardView) findViewById(R.id.monthCard);
        yearCard = (CardView) findViewById(R.id.yearCard);

        tvMonth = (TextView) findViewById(R.id.tvMonth);
        tvYear = (TextView) findViewById(R.id.tvYear);


tvMonth.setText(months.get(Integer.parseInt(prefs.getString("month",currentMonth +
"")+1)));
        tvYear.setText(prefs.getString("year", currentYear + ""));

        monthCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Toast.makeText(MainActivity.this,"Index of: " +
prefs.getString("month",currentMonth + ""), Toast.LENGTH_LONG).show();
                Intent intent = new
Intent(MainActivity.this,com.example.ptyxiakh2.SelectMonthActivity.class);
                intent.putExtra("month",prefs.getString("month",currentMonth + ""));
                startActivityForResult(intent, SELECT_MONTH_ACTIVITY);
            }
        });

        yearCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(MainActivity.this,com.example.ptyxiakh2.SelectYearActivity.class);
                intent.putExtra("year",prefs.getString("year",currentYear + ""));
```

```
            intent.putExtra("year_zero",          year_zero          +          "");
            startActivityForResult(intent,                    SELECT_YEAR_ACTIVITY);
        }
    });

    tvMonthlyDBData      =      (TextView)      findViewById(R.id.tvMonthlyDBData);
    tvMonthlyKWh         =       (TextView)        findViewById(R.id.tvMonthlyKWh);
    tvMonthlyCost        =       (TextView)       findViewById(R.id.tvMonthlyCost);
    tvMonthlyMaxPower     =     (TextView)     findViewById(R.id.tvMonthlyMaxPower);

    tvMonthlyDBData.setText(prefs.getString("tvMonthlyDBData",             "N/A"));
    tvMonthlyKWh.setText(prefs.getString("tvMonthlyKWh",                   "N/A"));
    tvMonthlyCost.setText(prefs.getString("tvMonthlyCost",                 "N/A"));
    tvMonthlyMaxPower.setText(prefs.getString("tvMonthlyMaxPower",         "N/A"));

    tvVmax             =            (TextView)            findViewById(R.id.tvVmax);
    tvVmin             =            (TextView)            findViewById(R.id.tvVmin);
    tvImax             =            (TextView)            findViewById(R.id.tvImax);
    tvImin             =            (TextView)            findViewById(R.id.tvImin);
    tvfMax             =            (TextView)            findViewById(R.id.tvfMax);
    tvfMin             =            (TextView)            findViewById(R.id.tvfMin);
    tvPfMax            =           (TextView)           findViewById(R.id.tvPfMax);
    tvPfMin            =           (TextView)           findViewById(R.id.tvPfMin);

    tvVmax.setText(prefs.getString("tvVmax",                               "N/A"));
    tvVmin.setText(prefs.getString("tvVmin",                               "N/A"));
    tvImax.setText(prefs.getString("tvImax",                               "N/A"));
    tvImin.setText(prefs.getString("tvImin",                               "N/A"));
    tvfMax.setText(prefs.getString("tvfMax",                               "N/A"));
    tvfMin.setText(prefs.getString("tvfMin","N/A"));
    tvPfMax.setText(prefs.getString("tvPfMax",                             "N/A"));
    tvPfMin.setText(prefs.getString("tvPfMin",                             "N/A"));

    actionBar                    =                    getSupportActionBar();

    if(prefs.getString("first_run","yes").equals("yes"))                        {
        SharedPreferences.Editor      editor      =      (SharedPreferences.Editor)
prefs.edit();
```

```
            editor.putString("kWh_cost",                                    kWh_cost);
            editor.putString("month",           currentMonth          +          "");
            editor.putString("year",            currentYear           +          "");
            editor.putString("date",                                       today);
            editor.putString("time",                                         now);
            editor.putString("url_single",                                  url1);
            editor.putString("url_all",url2);
            editor.putString("update",                                     "yes");
            editor.putString("first_run",                                   "no");
            editor.commit();
            Toast.makeText(MainActivity.this, "Please go to settings and set your
username/password!",Toast.LENGTH_LONG).show();
        }

        //Toast.makeText(MainActivity.this,    "To    update?    (on_create)    "    +
prefs.getString("update","yes"),Toast.LENGTH_LONG).show();

    }

    @Override
    public          boolean          onCreateOptionsMenu(Menu          menu)          {
        getMenuInflater().inflate(R.menu.menu,                                  menu);
        MenuItem          item1          =          menu.findItem(R.id.refresh_data);
        if                    (item1                    !=                    null)
            item1.setVisible(true);

        MenuItem            item2            =            menu.findItem(R.id.settings);
        if                (item2                !=                null)                {
            item2.setVisible(true);
        }

        MenuItem          item3          =          menu.findItem(R.id.datetimePicker);
        if                (item3                !=                null)                {
            item3.setVisible(true);
        }

        MenuItem            item4            =            menu.findItem(R.id.search);
        if                (item4                !=                null)                {
```

```
            item4.setVisible(false);
        }
        return                                          super.onCreateOptionsMenu(menu);
    }

    @Override
    public     boolean     onOptionsItemSelected(@NonNull     MenuItem     item)     {

        if(item.getItemId()                  ==                  R.id.refresh_data){
            if(prefs.getString("username","").equals("")                              ||
prefs.getString("password","").equals("")){
                Toast.makeText(MainActivity.this, "Please go to settings and set your
username/password!",Toast.LENGTH_LONG).show();
            }
            else                                                                      {
                //new                                          GetData().execute(url1);
                String        []        urlArray        =        new        String[1];
                urlArray[0]=                                                    url1;
                getData                 =                 new                 GetData();
                getData.execute(urlArray);
                //Toast.makeText(MainActivity.this,     "Update     TextViews     "     +
prefs.getString("tvUpdateTime","N/A"),Toast.LENGTH_LONG).show();
            }
        }
        else            if(item.getItemId()            ==            R.id.settings){
            Intent                    intent                    =                    new
Intent(MainActivity.this,com.example.ptyxiakh2.SettingsActivity.class);
            intent.putExtra("username",       prefs.getString("username",       ""));
            intent.putExtra("password",       prefs.getString("password",       ""));
            intent.putExtra("kWh_cost",    prefs.getString("kWh_cost",    kWh_cost));
            startActivityForResult(intent,                          SETTINGS_ACTIVITY);
        }
        else          if(item.getItemId()          ==          R.id.datetimePicker){
            if(prefs.getString("username","").equals("")                              ||
prefs.getString("password","").equals("")){
                Toast.makeText(MainActivity.this, "Please go to settings and set your
username/password!",Toast.LENGTH_LONG).show();
            }
            else                                                                      {
```

```
                //Toast.makeText(MainActivity.this,         "Year:         "         +
prefs.getString("year",currentYear          +          ""),Toast.LENGTH_LONG).show();
                Intent       intent       =       new       Intent(MainActivity.this,
com.example.ptyxiakh2.DatetimePickerActivity.class);
                intent.putExtra("date",        prefs.getString("date",        today));
                intent.putExtra("time",         prefs.getString("time",         now));
                intent.putExtra("month",   prefs.getString("month",   currentMonth   +
""));
                intent.putExtra("year",  prefs.getString("year",  currentYear  +  ""));
                intent.putExtra("year_zero",          year_zero          +          "");
                intent.putExtra("currentYear",        currentYear        +        "");
                intent.putExtra("username",   prefs.getString("username",   username));
                intent.putExtra("password",   prefs.getString("password",   password));

                startActivity(intent);
            }
        }

        return                                    super.onOptionsItemSelected(item);

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data)                                                                              {
        super.onActivityResult(requestCode,             resultCode,             data);
        if(requestCode                      ==                    SETTINGS_ACTIVITY){
            if(resultCode                          ==                          RESULT_OK){
                username              =              data.getStringExtra("username");
                password              =              data.getStringExtra("password");
                if(!data.getStringExtra("kWh_cost").toString().trim().isEmpty())
                    kWh_cost            =            data.getStringExtra("kWh_cost");
                SharedPreferences.Editor    editor    =    (SharedPreferences.Editor)
prefs.edit();
                editor.putString("kWh_cost",                                kWh_cost);
                editor.putString("username",                                username);
                editor.putString("password",                                password);
                editor.putString("update",                                    "yes");
                editor.commit();
```

```
                //Toast.makeText(MainActivity.this,"The KWh Cost is now: " +
kWh_cost,                                                    Toast.LENGTH_LONG).show();
            }
            else            if(resultCode            ==            RESULT_CANCELED){
            }
        }
        else            if(requestCode            ==            SELECT_YEAR_ACTIVITY){
            if(resultCode                        ==                        RESULT_OK){
                String          year          =          data.getStringExtra("year");
                SharedPreferences.Editor    editor    =    (SharedPreferences.Editor)
prefs.edit();
                if(!prefs.getString("year",currentYear    +    "").equals(year))    {
                    editor.putString("year",                                    year);
                    editor.putString("update",                                  "yes");
                }
                editor.commit();

               //Toast.makeText(MainActivity.this,    "To    update?    (year)    "    +
prefs.getString("update","yes"),Toast.LENGTH_LONG).show();
                tvYear.setText(prefs.getString("year",     currentYear     +     ""));
            }
            else            if(resultCode            ==            RESULT_CANCELED){
            }
        }
        else            if(requestCode            ==            SELECT_MONTH_ACTIVITY){
            if(resultCode                        ==                        RESULT_OK){
                String          month          =          data.getStringExtra("month");
                SharedPreferences.Editor    editor    =    (SharedPreferences.Editor)
prefs.edit();
                if(!prefs.getString("month",currentMonth    +    "").equals(month))    {
                    editor.putString("month",                                   month);
                    editor.putString("update",                                  "yes");
                }
                editor.commit();

                //Toast.makeText(MainActivity.this,    "To    update?    (month)    "    +
prefs.getString("update","yes"),Toast.LENGTH_LONG).show();

tvMonth.setText(months.get(Integer.parseInt(prefs.getString("month",currentMonth     +
```

```
""))));
            }
            else if(resultCode == RESULT_CANCELED){
            }
        }
    }

    public class GetData extends AsyncTask<String, Integer, ArrayList<String>> {

        ProgressDialog dialog;

        private Exception exception;
        JSONParser jParser = new JSONParser();

        boolean userCancelled = false;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();

            dialog = new ProgressDialog(MainActivity.this);
            dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            dialog.setCancelable(true);
            dialog.show();

            dialog.setOnCancelListener(new DialogInterface.OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog1) {
                    getData.cancel(true);
                    userCancelled = true;
                    dialog.dismiss();
                    //Toast.makeText(MainActivity.this,"AsyncTask is
stopped",Toast.LENGTH_LONG).show();
                }
            });

            new CountDownTimer(5000, 1000) {
```

```
            @Override
            public void onTick(long millisUntilFinished) {
                //Toast.makeText(MainActivity.this,"Waiting for
data...",Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onFinish() {
                if (getData.getStatus() == AsyncTask.Status.RUNNING) {
                    //Toast.makeText(MainActivity.this,"Waiting for
data...",Toast.LENGTH_SHORT).show();
                    getData.cancel(true);
                    dialog.dismiss();
                }
            }
        }.start();
    }

    protected ArrayList<String> doInBackground(String... urls) {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        ArrayList<String> jsonString = new ArrayList<>();
        int count = urls.length;
        for (int i = 0; i < count; i++) {
            publishProgress((int) ((i / (float) count) * 100));
            if(getData.isCancelled()) break;
            try {
                params.add(new BasicNameValuePair("email",
prefs.getString("username", "")));
                params.add(new BasicNameValuePair("password",
prefs.getString("password", "")));
                params.add(new BasicNameValuePair("month",
prefs.getString("month", currentMonth + "")));
                params.add(new BasicNameValuePair("year", prefs.getString("year",
currentYear + "")));
                params.add(new BasicNameValuePair("update",
prefs.getString("update", "yes")));

                JSONObject json = jParser.makeHttpRequest(urls[i].toString(),
"POST", params);
```

```
                //Log.d("JSON:                                    ",json.toString());
                jsonString.add(json.toString());
            }           catch           (Exception           e)           {
                this.exception                          =                       e;
                return                                                      null;
            }
        }

        return                                                          jsonString;
    }

    @Override
    protected       void       onProgressUpdate(Integer...       values)       {
        super.onProgressUpdate(values);
        dialog.setProgress(values[0]);
    }

    @Override
    protected       void       onPostExecute(ArrayList<String>       s)       {
        super.onPostExecute(s);
        dialog.dismiss();

        JSONObject      json,      values,      data,      total,      monthly;
        String update_date = "N/A", active_energy_in = "N/A", active_energy_out =
"N/A",                  active_power                        =                       "N/A";
        String voltage = "N/A", current = "N/A", frequency = "N/A", power_factor
=                                                                               "N/A";
        String total_db_data = "N/A", total_max_power = "N/A", init_db_date =
"N/A";
        String update_date_date = "N/A", update_date_time = "N/A", total_cost =
"N/A";
        String monthly_db_data = "N/A", month_max_power = "N/A", month_kWh =
"N/A",                  month_cost                      =                       "N/A";
        String imax = "N/A",  imin = "N/A", vmax = "N/A", vmin = "N/A", fmax =
"N/A",    fmin    =    "N/A",    pfmax    =    "N/A",    pfmin    =    "N/A";

        for(String          jsondata          :          s)          {
            try                                                             {
                json          =          new          JSONObject(jsondata);
```

```
if (Integer.parseInt(json.getString("value")) == 1) {
    values = json.getJSONObject("values");
    update_date = values.getString("date");

    data = values.getJSONObject("data");

    Iterator<String> iter = data.keys();
    while (iter.hasNext()) {
        String key = iter.next();
        try {
            if (key.equals("active_energy_in")) {
                if(!data.getString(key).equals("-1"))
                    active_energy_in = data.getString(key);
            } else if (key.equals("active_energy_out")) {
                if(!data.getString(key).equals("-1"))
                    active_energy_out = data.getString(key);
            } else if (key.equals("active_power")) {
                if(!data.getString(key).equals("-1"))
                    active_power = data.getString(key);
            } else if (key.equals("voltage")) {
                if(!data.getString(key).equals("-1"))
                    voltage = data.getString(key);
            } else if (key.equals("current")) {
                if(!data.getString(key).equals("-1"))
                    current = data.getString(key);
            } else if (key.equals("frequency")) {
                if(!data.getString(key).equals("-1"))
                    frequency = data.getString(key);
            } else if (key.equals("power_factor")) {
                if(!data.getString(key).equals("-1"))
                    power_factor = data.getString(key);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    total = values.getJSONObject("total");
```

```
//Log.d("Total:                ",              total.toString());
Iterator<String>          iter1          =          total.keys();

while                      (iter1.hasNext())                    {
    String             key             =             iter1.next();
    try                                                         {
        if            (key.equals("total_db_data"))            {
            if(!total.getString(key).equals("-1"))
                total_db_data      =      total.getString(key);
        }    else    if    (key.equals("total_max_power"))    {
            if(!total.getString(key).equals("-1"))
                total_max_power     =     total.getString(key);
        }     else     if     (key.equals("init_db_date"))     {
            if(!total.getString(key).equals("-1"))
                init_db_date      =      total.getString(key);
        }
    }         catch         (JSONException         e)         {
        e.printStackTrace();
    }
}

if    (prefs.getString("update",    "yes").equals("yes"))    {

    monthly          =          values.getJSONObject("monthly");
    //Log.d("Total:             ",             total.toString());
    Iterator<String>       iter2       =       monthly.keys();

    while                    (iter2.hasNext())                  {
        String           key           =           iter2.next();
        try                                                     {
            if         (key.equals("monthly_db_data"))         {
                if(!monthly.getString(key).equals("-1"))
                    monthly_db_data  =  monthly.getString(key);
            }   else   if   (key.equals("month_max_power"))   {
                if(!monthly.getString(key).equals("-1"))
                    month_max_power  =  monthly.getString(key);
            }     else     if     (key.equals("month_kWh"))     {
                if(!monthly.getString(key).equals("-1"))
                    month_kWh     =     monthly.getString(key);
```

```
                }
                else if (key.equals("kwh_missed")) {
                    if(!monthly.getString(key).equals("-1"))
                        month_kWh = Integer.parseInt(month_kWh) +
Integer.parseInt(monthly.getString(key)) + "";
                }
                else if (key.equals("imax")) {
                    if(!monthly.getString(key).equals("-1"))
                        imax = monthly.getString(key);
                }
                else if (key.equals("imin")) {
                    if(!monthly.getString(key).equals("-1"))
                        imin = monthly.getString(key);
                }
                else if (key.equals("vmax")) {
                    if(!monthly.getString(key).equals("-1"))
                        vmax = monthly.getString(key);
                }
                else if (key.equals("vmin")) {
                    if(!monthly.getString(key).equals("-1"))
                        vmin = monthly.getString(key);
                }
                else if (key.equals("fmax")) {
                    if(!monthly.getString(key).equals("-1"))
                        fmax = monthly.getString(key);
                }
                else if (key.equals("fmin")) {
                    if(!monthly.getString(key).equals("-1"))
                        fmin = monthly.getString(key);
                }
                else if (key.equals("pfmax")) {
                    if(!monthly.getString(key).equals("-1"))
                        pfmax = monthly.getString(key);
                }
                else if (key.equals("pfmin")) {
                    if(!monthly.getString(key).equals("-1"))
                        pfmin = monthly.getString(key);
                }
            } catch (JSONException e) {
                e.printStackTrace();
```

```
                }
            }

            Double cost_m;
            cost_m = 0.0;
            if(!month_kWh.equals("N/A")) {
                cost_m = Double.parseDouble(prefs.getString("kWh_cost", kWh_cost)) *
Double.parseDouble(month_kWh);
            }
            DecimalFormat df = new DecimalFormat("####0.00");
            month_cost = df.format(cost_m) ;

            SharedPreferences.Editor editor =
(SharedPreferences.Editor) prefs.edit();
            if(!prefs.getString("month", currentMonth +
"").equals(currentMonth + "")) {
                editor.putString("update", "no");
            }
            editor.putString("tvMonthlyDBData",monthly_db_data);
            editor.putString("tvMonthlyKWh",month_kWh);
            editor.putString("tvMonthlyCost", month_cost);
            editor.putString("tvMonthlyMaxPower", month_max_power);
            editor.putString("tvVmax", vmax);
            editor.putString("tvVmin", vmin);
            editor.putString("tvImax", imax);
            editor.putString("tvImin", imin);
            editor.putString("tvfMax", fmax);
            editor.putString("tvfMin", fmin);
            editor.putString("tvPfMax", pfmax);
            editor.putString("tvPfMin", pfmin);
            editor.commit();


tvMonthlyDBData.setText(prefs.getString("tvMonthlyDBData", "N/A"));
            tvMonthlyKWh.setText(prefs.getString("tvMonthlyKWh",
"N/A"));
            tvMonthlyCost.setText(prefs.getString("tvMonthlyCost",
"N/A"));
```

```
tvMonthlyMaxPower.setText(prefs.getString("tvMonthlyMaxPower",                "N/A"));

                        tvVmax.setText(prefs.getString("tvVmax",          "N/A"));
                        tvVmin.setText(prefs.getString("tvVmin",          "N/A"));
                        tvImax.setText(prefs.getString("tvImax",          "N/A"));
                        tvImin.setText(prefs.getString("tvImin",          "N/A"));
                        tvfMax.setText(prefs.getString("tvfMax",          "N/A"));
                        tvfMin.setText(prefs.getString("tvfMin","N/A"));
                        tvPfMax.setText(prefs.getString("tvPfMax",        "N/A"));
                        tvPfMin.setText(prefs.getString("tvPfMin",        "N/A"));

                        //Toast.makeText(MainActivity.this, "To update? (refresh)
"           +           prefs.getString("update","yes"),Toast.LENGTH_LONG).show();
                    }

                    Double                                                  cost;
                    cost                        =                           0.0;
                    if(!active_energy_in.equals("N/A"))                        {
                        cost    =    Double.parseDouble(prefs.getString("kWh_cost",
kWh_cost))                      *                   Double.parseDouble(active_energy_in);
                    }
                    DecimalFormat    df    =    new    DecimalFormat("####0.00");
                    total_cost          =          df.format(cost)          ;

                    if(!update_date.equals("N/A"))                             {
                        String[]    date_parts    =    update_date.split("    ");
                        update_date_date            =            date_parts[0];
                        update_date_time            =            date_parts[1];
                    }

                    SharedPreferences.Editor  editor  =  (SharedPreferences.Editor)
prefs.edit();
                    editor.putString("tvUpdateDate",          update_date_date);
                    editor.putString("tvActiveEnergyIn",      active_energy_in);
                    editor.putString("tvUpdateTime",          update_date_time);
                    editor.putString("tvActivePower",             active_power);
                    editor.putString("tvVoltage",voltage);
                    editor.putString("tvCurrent",                     current);
                    editor.putString("tvFrequency",                 frequency);
```

```
                    editor.putString("tvPF",                    power_factor);
                    editor.putString("tvStartDate",             init_db_date);
                    editor.putString("tvTotalDBData",           total_db_data);
                    editor.putString("tvTotalKWh",           active_energy_in);
                    editor.putString("tvTotalCost",                total_cost);
                    editor.putString("tvMaxPower",            total_max_power);
                    editor.commit();

                    tvUpdateDate.setText(prefs.getString("tvUpdateDate","N/A"));
                    tvActiveEnergyIn.setText(prefs.getString("tvActiveEnergyIn",
"N/A"));
                    tvUpdateTime.setText(prefs.getString("tvUpdateTime",  "N/A"));
                    tvActivePower.setText(prefs.getString("tvActivePower",
"N/A"));
                    tvVoltage.setText(prefs.getString("tvVoltage",        "N/A"));
                    tvCurrent.setText(prefs.getString("tvCurrent",        "N/A"));
                    tvFrequency.setText(prefs.getString("tvFrequency",    "N/A"));
                    tvPF.setText(prefs.getString("tvPF","N/A"));

                    tvStartDate.setText(prefs.getString("tvStartDate",    "N/A"));
                    tvTotalDBData.setText(prefs.getString("tvTotalDBData",
"N/A"));
                    tvTotalKWh.setText(prefs.getString("tvTotalKWh",      "N/A"));
                    tvTotalCost.setText(prefs.getString("tvTotalCost",    "N/A"));
                    tvMaxPower.setText(prefs.getString("tvMaxPower",      "N/A"));

                }
                else{
                    Toast.makeText(MainActivity.this,json.getString("message")
,Toast.LENGTH_LONG).show();
                }
            }           catch           (JSONException           e)           {
                e.printStackTrace();
            }
        }

        //Toast.makeText(MainActivity.this, "Update TextViews (async task) " +
prefs.getString("tvUpdateTime","N/A")     +     "     update_date_time:     "     +
update_date_time,Toast.LENGTH_LONG).show();
```

```
        }

        @Override
        protected void onCancelled() {
            if(userCancelled)Toast.makeText(MainActivity.this, "Update Cancelled by
user",Toast.LENGTH_LONG).show();
            else Toast.makeText(MainActivity.this, "Server taking too long to
respond",Toast.LENGTH_LONG).show();
            super.onCancelled();
        }
    }
}
```

SettingsActivity.java

```
package com.example.ptyxiakh2;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import org.w3c.dom.Text;

public class SettingsActivity extends AppCompatActivity {

    EditText tvUsername, tvPassword, tvKWhCost;

    Button btnCancel, btnUpdate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_settings);

        tvUsername          =          (EditText)          findViewById(R.id.tvUsername);
        tvPassword          =          (EditText)          findViewById(R.id.tvPassword);
        tvKWhCost           =          (EditText)           findViewById(R.id.tvkWhCost);

        tvUsername.setText(getIntent().getStringExtra("username"));
        tvPassword.setText(getIntent().getStringExtra("password"));
        tvKWhCost.setText(getIntent().getStringExtra("kWh_cost"));

        btnCancel           =           (Button)           findViewById(R.id.btnCancel);
        btnUpdate           =           (Button)           findViewById(R.id.btnUpdate);

        btnCancel.setOnClickListener(new            View.OnClickListener()            {
            @Override
            public             void             onClick(View             v)             {
                setResult(RESULT_CANCELED);
                SettingsActivity.this.finish();
            }
        });

        btnUpdate.setOnClickListener(new            View.OnClickListener()            {
            @Override
            public             void             onClick(View             v)             {
                Intent            intent            =            new            Intent();
                intent.putExtra("username",   tvUsername.getText().toString().trim());
                intent.putExtra("password",   tvPassword.getText().toString().trim());
                intent.putExtra("kWh_cost",tvKWhCost.getText().toString().trim());
                setResult(RESULT_OK,                                                intent);
                SettingsActivity.this.finish();
            }
        });
    }
}
```

JSONParser.java

```
package com.example.ptyxiakh2;

import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;

public class JSONParser {
    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = "";

    // constructor
    public JSONParser() { }

    // function get json from url
    // by making HTTP POST or GET mehtod
    public JSONObject makeHttpRequest(String url, String method, List<NameValuePair>
params) {

        // Making HTTP request
        try {
            // check for request method
```

```
    if(method == "POST"){
        // request method is POST
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost(url);
        httpPost.setEntity(new UrlEncodedFormEntity(params));

        HttpResponse httpResponse = httpClient.execute(httpPost);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();

    }else if(method == "GET"){
        // request method is GET
        DefaultHttpClient httpClient = new DefaultHttpClient();
        String paramString = URLEncodedUtils.format(params, "utf-8");
        url += "?" + paramString;
        HttpGet httpGet = new HttpGet(url);

        HttpResponse httpResponse = httpClient.execute(httpGet);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
        //Log.d("GET METHOD:", "GET " + is.toString());
    }

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

try {
    BufferedReader reader = new BufferedReader(new InputStreamReader(
            is, "UTF-8"), 8);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }
    is.close();
```

```
            json = sb.toString();
            //Log.d("Json", "Json: " + json);
        } catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try parse the string to a JSON object
        try {
            jObj = new JSONObject(json);
            //Log.d("Json_obj", "Json_obj: " + jObj.toString());
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

        // return JSON String
        return jObj;

    }
}
```

DatetimePickerActivity.java

```
package com.example.ptyxiakh2;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.view.MenuItemCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.DatePickerDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
```

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.SearchView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.lang.reflect.Array;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;

public class DatetimePickerActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener,              EnergyDataAdapter.EnergyDataClicked{
```

```
    ActionBar                                                    actionBar;

    RecyclerView                                                 list;
    RecyclerView.LayoutManager                                   layoutManager;
    EnergyDataAdapter                                            energyDataAdapter;

    TextView tvUpdateDate, tvActiveEnergyIn, tvUpdateTime, tvActivePower, tvVoltage,
tvCurrent,                          tvFrequency,                          tvPF;
    EditText                                                     day;

    int                                                          my_day;

    Spinner                                                      monthChooser;

    String[] months = { "January", "February", "March", "April", "May", "June",
"July",     "August",     "September",     "October",     "November",     "December"};

    Spinner                                                      yearChooser;

    Integer[]                                                    years;

    String                              year,                    month;

    String                          username,                    password;

    String  url2  =  "http://www.moijoune.net/dissertation/readalldata_android1.php";

    AsyncTask                   getData                  =                  null;

    ArrayList<EnergyData>      energyDataArrayList      =      new      ArrayList<>();

    @Override
    protected        void        onCreate(Bundle        savedInstanceState)        {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_datetime_picker);

        tvUpdateDate        =        (TextView)        findViewById(R.id.tvUpdateDate);
```

```
        tvActiveEnergyIn = (TextView) findViewById(R.id.tvActiveEnergyIn);
        tvUpdateTime = (TextView) findViewById(R.id.tvUpdateTime);
        tvActivePower = (TextView) findViewById(R.id.tvActivePower);
        tvVoltage = (TextView) findViewById(R.id.tvVoltage);
        tvCurrent = (TextView) findViewById(R.id.tvCurrent);
        tvFrequency = (TextView) findViewById(R.id.tvFrequency);
        tvPF = (TextView) findViewById(R.id.tvPF);

        list = findViewById(R.id.list);
        list.setHasFixedSize(true);

        //Toast.makeText(MainActivity.this, "WELCOME!",Toast.LENGTH_LONG).show();

        layoutManager = new LinearLayoutManager(this);

        list.setLayoutManager(layoutManager);

        day = (EditText) findViewById(R.id.day);

        day.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                day.setCursorVisible(true);
            }
        });

        monthChooser = (Spinner) findViewById(R.id.monthChooser);
        monthChooser.setOnItemSelectedListener(this);

        ArrayAdapter monthChooserAdapter = new
ArrayAdapter(this,android.R.layout.simple_spinner_item,months);

monthChooserAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        monthChooser.setAdapter(monthChooserAdapter);

        month = getIntent().getStringExtra("month");
        monthChooser.setSelection(Integer.parseInt(month));
```

```
        int year_zero = Integer.parseInt(getIntent().getStringExtra("year_zero"));
        int current_year = Integer.parseInt(getIntent().getStringExtra("currentYear"));

        int length = current_year - year_zero + 1;

        years = new Integer[length];

        //Toast.makeText(DatetimePickerActivity.this, "Year_zero: " + year_zero + ", Current_year: " + current_year, Toast.LENGTH_LONG).show();

        for(int i = year_zero; i < current_year + 1; i++){
            years[i - year_zero] = i;
        }

        yearChooser = (Spinner) findViewById(R.id.yearChooser);
        yearChooser.setOnItemSelectedListener(this);

        ArrayAdapter yearChooserAdapter = new ArrayAdapter(this,android.R.layout.simple_spinner_item,years);

yearChooserAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        yearChooser.setAdapter(yearChooserAdapter);

        year = getIntent().getStringExtra("year");
        //Toast.makeText(DatetimePickerActivity.this, "Year from MainActivity: " + year + ". Index: " + Arrays.asList(years).indexOf(Integer.parseInt(year)), Toast.LENGTH_LONG).show();

yearChooser.setSelection(Arrays.asList(years).indexOf(Integer.parseInt(year)));

        actionBar = getSupportActionBar();

        username = getIntent().getStringExtra("username");
        password = getIntent().getStringExtra("password");
```

```
        energyDataArrayList.clear();

        energyDataAdapter = new EnergyDataAdapter(DatetimePickerActivity.this,
energyDataArrayList);
        list.setAdapter(energyDataAdapter);

        //Toast.makeText(DatetimePickerActivity.this, "Username: " + username + ".
Password: " + password, Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);

        MenuItem item1 = menu.findItem(R.id.search);
        if (item1 != null) {
            SearchView searchView = (SearchView) MenuItemCompat.getActionView(item1);
            searchView.setOnQueryTextListener(
                    new SearchView.OnQueryTextListener() {

                        @Override
                        public boolean onQueryTextSubmit(String query)
                        {
                            return false;
                        }

                        @Override
                        public boolean onQueryTextChange(String newText)
                        {
                            energyDataAdapter.getFilter().filter(newText);
                            return false;
                        }
                    });
        }

        MenuItem item2 = menu.findItem(R.id.refresh_data);
        if (item2 != null)
            item2.setVisible(true);
```

```
        MenuItem item3 = menu.findItem(R.id.settings);
        if (item3 != null) {
            item3.setVisible(false);
        }

        MenuItem item4 = menu.findItem(R.id.datetimePicker);
        if (item4 != null) {
            item4.setVisible(false);
        }
        return super.onCreateOptionsMenu(menu);
    }


    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {

        if(item.getItemId() == R.id.refresh_data){
            day.setCursorVisible(false);
            Calendar c = Calendar.getInstance();
            c.set(Integer.parseInt(year), Integer.parseInt(month), 1);
            int day_max = c.getActualMaximum(Calendar.DAY_OF_MONTH);
            int day_min = 1;

            if(day.getText().toString().isEmpty()) {
                my_day = 0;
            }
            else{
                my_day = Integer.parseInt(day.getText().toString().trim());
            }

            if(my_day > day_max || my_day < day_min)
                Toast.makeText(DatetimePickerActivity.this, "Not a valid date! Please
choose another!", Toast.LENGTH_LONG).show();
            else{
                //Toast.makeText(DatetimePickerActivity.this, "Date Chosen: " +
my_day + " " + months[Integer.parseInt(month)] + " " + year,
Toast.LENGTH_LONG).show();
```

```
                //call                              async                              task
                String        []        urlArray        =        new        String[1];
                urlArray[0]=                                                        url2;
                getData                =                new                GetData();
                getData.execute(urlArray);
                //new                                        GetData().execute(url2);
            }
        }

        return                                      super.onOptionsItemSelected(item);

    }

    @Override
    public        void        onEnergyDataClicked(EnergyData        energyData)        {
        Intent    intent    =    new    Intent(this,com.example.ptyxiakh2.ShowData.class);
        intent.putExtra("update_date",energyData.getUpdate_date());
        intent.putExtra("active_energy_in",energyData.getActive_energy_in());
        intent.putExtra("active_power",energyData.getActive_power());
        intent.putExtra("voltage",                              energyData.getVoltage());
        intent.putExtra("current",                              energyData.getCurrent());
        intent.putExtra("frequency",                          energyData.getFrequency());
        intent.putExtra("power_factor",                  energyData.getPower_factor());

        startActivity(intent);
    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long
id)                                                                                 {
        switch                                                    (parent.getId()){
            case                                                 R.id.monthChooser:
                month              =              position              +              "";
                //Toast.makeText(DatetimePickerActivity.this,"Month    Chosen:    "    +
months[position],                                          Toast.LENGTH_LONG).show();
                break;
            case                                                 R.id.yearChooser:
                year            =            years[position]            +            "";
                //Toast.makeText(DatetimePickerActivity.this,"Year    Chosen:    "    +
```

```
years[position],                                    Toast.LENGTH_LONG).show();
                break;
            }
        }

        @Override
        public        void        onNothingSelected(AdapterView<?>        parent)        {
            switch                          (parent.getId())                          {
                case                                                  R.id.monthChooser:
                    Toast.makeText(DatetimePickerActivity.this, "Please select a month!",
Toast.LENGTH_LONG).show();
                break;
                case                                                  R.id.yearChooser:
                    Toast.makeText(DatetimePickerActivity.this, "Please select a year!",
Toast.LENGTH_LONG).show();
                //Toast.makeText(DatetimePickerActivity.this,"Year    Chosen:    "    +
years[position],                                    Toast.LENGTH_LONG).show();
                break;
            }
        }

        public class GetData extends AsyncTask<String, Integer, ArrayList<String>> {

            ProgressDialog                                                  dialog;

            private                         Exception                         exception;
            JSONParser          jParser          =          new          JSONParser();

            boolean                userCancelled                =                false;

            @Override
            protected                void                onPreExecute()                {
                super.onPreExecute();
                energyDataArrayList.clear();

                dialog      =      new      ProgressDialog(DatetimePickerActivity.this);
                dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                dialog.setCancelable(true);
```

```
            dialog.show();

            dialog.setOnCancelListener(new DialogInterface.OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog1) {
                    userCancelled = true;
                    getData.cancel(true);
                    dialog.dismiss();
                }
            });

            new CountDownTimer(5000, 1000) {

                @Override
                public void onTick(long millisUntilFinished) {
                    //Toast.makeText(MainActivity.this,"Waiting for
data...",Toast.LENGTH_SHORT).show();
                }

                @Override
                public void onFinish() {
                    if (getData.getStatus() == AsyncTask.Status.RUNNING) {
                        //Toast.makeText(MainActivity.this,"Waiting for
data...",Toast.LENGTH_SHORT).show();
                        getData.cancel(true);
                        dialog.dismiss();
                    }
                }
            }.start();

        }

        protected ArrayList<String> doInBackground(String... urls) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            ArrayList<String> jsonString = new ArrayList<>();
            int count = urls.length;
            for (int i = 0; i < count; i++) {
                publishProgress((int) ((i / (float) count) * 100));
```

```
            if (getData.isCancelled()) break;
            try {
                params.add(new BasicNameValuePair("email",username));
                params.add(new BasicNameValuePair("password", password));
                params.add(new BasicNameValuePair("month", month));
                params.add(new BasicNameValuePair("year", year));
                params.add(new BasicNameValuePair("day", my_day + ""));

                JSONObject json = jParser.makeHttpRequest(urls[i].toString(),
"POST", params);
                //Log.d("JSON: ",json.toString());
                jsonString.add(json.toString());
            } catch (Exception e) {
                this.exception = e;
                return null;
            }
        }

        return jsonString;
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        dialog.setProgress(values[0]);
    }

    @Override
    protected void onPostExecute(ArrayList<String> s) {
        super.onPostExecute(s);
        dialog.dismiss();
        EnergyData energyData;

        JSONObject json, datedata, data;
        JSONArray values;
        String json_data;
        String update_date = "N/A", active_energy_in = "N/A", active_energy_out =
"N/A", active_power = "N/A";
        String voltage = "N/A", current = "N/A", frequency = "N/A", power_factor
```

```
=                                                                    "N/A";

            for(String            jsondata            :            s)            {
                try                                                                {
                    json            =            new            JSONObject(jsondata);

                    if    (Integer.parseInt(json.getString("value"))    ==    1)    {
                        try                                                        {
                            values            =            json.getJSONArray("values");
                            for   (int   i   =   0;   i   <   values.length();   i++)   {
                                try                                                {
                                    datedata        =        values.getJSONObject(i);
                                    json_data       =       datedata.getString("date");
                                    if(!json_data.equals("-1"))
                                        update_date            =            json_data;
                                    try{
                                        data     =     datedata.getJSONObject("data");
                                        Iterator<String>    iter    =    data.keys();
                                        while            (iter.hasNext())            {
                                            String       key       =       iter.next();
                                            try                                    {
                                                if (key.equals("active_energy_in")) {
                                                    if(!data.getString(key).equals("-
1"))
                                                        active_energy_in            =
data.getString(key);
                                                } else if
(key.equals("active_power"))                                                       {
                                                    if(!data.getString(key).equals("-
1"))
                                                        active_power                =
data.getString(key);
                                                } else if (key.equals("voltage")) {
                                                    if(!data.getString(key).equals("-
1"))
                                                        voltage                     =
data.getString(key);
                                                } else if (key.equals("current")) {
                                                    if(!data.getString(key).equals("-
```

```
1"))
                                                    current                    =
data.getString(key);
                                                } else if (key.equals("frequency")) {
                                                    if(!data.getString(key).equals("-
1"))
                                                    frequency                  =
data.getString(key);
                                                }              else              if
(key.equals("power_factor"))                                                    {
                                                    if(!data.getString(key).equals("-
1"))
                                                    power_factor               =
data.getString(key);
                                                }
                                            }    catch    (JSONException    e)    {
                                                e.printStackTrace();
                                            }
                                        }
                                        energyData   =   new   EnergyData(update_date,
active_energy_in,  active_energy_out,  active_power,  voltage,  current,  frequency,
power_factor);
                                        energyDataArrayList.add(energyData);
                                    }
                                    catch             (JSONException             e){
                                        e.printStackTrace();
                                    }
                                }       catch       (JSONException       e)       {
                                    e.printStackTrace();
                                }
                            }
                        }
                        catch                   (JSONException                   e){
                            e.printStackTrace();
                        }
                    }
                    else{

Toast.makeText(DatetimePickerActivity.this,json.getString("message")
,Toast.LENGTH_LONG).show();
```

```
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }

        energyDataAdapter = new EnergyDataAdapter(DatetimePickerActivity.this, energyDataArrayList);
        list.setAdapter(energyDataAdapter);
        //energyDataAdapter.notifyDataSetChanged();
        //Log.d("0: ",energyDataArrayList.get(0).toString());
        //Toast.makeText(MainActivity.this, "Update TextViews (async task) " + prefs.getString("tvUpdateTime","N/A") + " update_date_time: " + update_date_time,Toast.LENGTH_LONG).show();

    }

    @Override
    protected void onCancelled() {
        if(userCancelled)Toast.makeText(DatetimePickerActivity.this, "Update Cancelled by user",Toast.LENGTH_LONG).show();
        else Toast.makeText(DatetimePickerActivity.this, "Server taking too long to respond",Toast.LENGTH_LONG).show();
        super.onCancelled();
    }
  }
}
```

ShowData.java

```
package com.example.ptyxiakh2;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.TextView;

public class ShowData extends AppCompatActivity {
```

```
    TextView tvUpdateDate, tvUpdateTime, tvActiveEnergyIn, tvActivePower, tvVoltage,
tvCurrent,                              tvFrequency,                              tvPF;

    @Override
    protected        void        onCreate(Bundle        savedInstanceState)        {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_show_data);

        tvUpdateDate        =        (TextView)        findViewById(R.id.tvUpdateDate);
        tvUpdateTime        =        (TextView)        findViewById(R.id.tvUpdateTime);
        tvActiveEnergyIn      =      (TextView)      findViewById(R.id.tvActiveEnergyIn);
        tvActivePower       =       (TextView)       findViewById(R.id.tvActivePower);
        tvVoltage          =          (TextView)          findViewById(R.id.tvVoltage);
        tvCurrent          =          (TextView)          findViewById(R.id.tvCurrent);
        tvFrequency         =         (TextView)         findViewById(R.id.tvFrequency);
        tvPF             =             (TextView)             findViewById(R.id.tvPF);

        if(!getIntent().getStringExtra("update_date").equals("N/A"))                {
            String[]  date_parts  =  getIntent().getStringExtra("update_date").split("
");
            String            update_date_date            =            date_parts[0];
            String            update_date_time            =            date_parts[1];
            tvUpdateDate.setText(update_date_date);
            tvUpdateTime.setText(update_date_time);
        }
        else{
            tvUpdateDate.setText("N/A");
            tvUpdateTime.setText("N/A");
        }

        tvActiveEnergyIn.setText(getIntent().getStringExtra("active_energy_in"));
        tvActivePower.setText(getIntent().getStringExtra("active_power"));
        tvVoltage.setText(getIntent().getStringExtra("voltage"));
        tvCurrent.setText(getIntent().getStringExtra("current"));
        tvFrequency.setText(getIntent().getStringExtra("frequency"));
        tvPF.setText(getIntent().getStringExtra("power_factor"));
    }
}
```

EnergyDataAdapter.java

```
package com.example.ptyxiakh2;

import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Filter;
import android.widget.Filterable;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
import java.util.List;

public class EnergyDataAdapter extends
RecyclerView.Adapter<EnergyDataAdapter.ViewHolder> implements Filterable {
    private ArrayList<EnergyData> energyDataList, tempItems, suggestions;

    EnergyDataClicked activity;

    public interface EnergyDataClicked{
        public void onEnergyDataClicked(EnergyData energyData);
    }

    public EnergyDataAdapter(Context context, ArrayList<EnergyData> energyDataList) {
        this.energyDataList = energyDataList;
        tempItems = new ArrayList<EnergyData>(energyDataList); // this makes the
difference.
        suggestions = new ArrayList<EnergyData>();
        activity = (EnergyDataClicked) context;
    }
```

```
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView tvUpdateDate, tvUpdateTime, tvActiveEnergyIn, tvActivePower,
tvVoltage, tvCurrent, tvFrequency, tvPF;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);

            tvUpdateDate = itemView.findViewById(R.id.tvUpdateDate);
            tvUpdateTime = itemView.findViewById(R.id.tvUpdateTime);
            tvActiveEnergyIn = itemView.findViewById(R.id.tvActiveEnergyIn);
            tvActivePower = itemView.findViewById(R.id.tvActivePower);
            tvVoltage = itemView.findViewById(R.id.tvVoltage);
            tvCurrent = itemView.findViewById(R.id.tvCurrent);
            tvFrequency = itemView.findViewById(R.id.tvFrequency);
            tvPF = itemView.findViewById(R.id.tvPF);

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    activity.onEnergyDataClicked((EnergyData) v.getTag());
                }
            });
        }
    }

    @NonNull
    @Override
    public EnergyDataAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_view,
parent, false);
        return new ViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull EnergyDataAdapter.ViewHolder holder, int
position) {
```

```
        holder.itemView.setTag(energyDataList.get(position));

holder.tvActiveEnergyIn.setText(energyDataList.get(position).getActive_energy_in());
        holder.tvActivePower.setText(energyDataList.get(position).getActive_power());
        holder.tvVoltage.setText(energyDataList.get(position).getVoltage());
        holder.tvCurrent.setText(energyDataList.get(position).getCurrent());
        holder.tvFrequency.setText(energyDataList.get(position).getFrequency());
        holder.tvPF.setText(energyDataList.get(position).getPower_factor());
        if(!energyDataList.get(position).getUpdate_date().equals("N/A"))            {
            String[]                            date_parts                          =
energyDataList.get(position).getUpdate_date().split("                           ");
            String            update_date_date            =            date_parts[0];
            String            update_date_time            =            date_parts[1];
            holder.tvUpdateDate.setText(update_date_date);
            holder.tvUpdateTime.setText(update_date_time);
        }
        else{
            holder.tvUpdateDate.setText("N/A");
            holder.tvUpdateTime.setText("N/A");
        }
    }

    @Override
    public                              Filter                              getFilter(){
        return                                                  getEnergyDataFilter;
    }

    Filter          getEnergyDataFilter          =          new          Filter(){

        @Override
        public     CharSequence     convertResultToString(Object     resultValue)     {
            String     str     =     ((EnergyData)     resultValue).getUpdate_date();
            return                                                              str;
        }

        @Override
        protected    FilterResults    performFiltering(CharSequence    constraint)    {
            if             (constraint             !=             null)             {
```

```
                suggestions.clear();
                //Log.e("Filter Results Count: ", suggestions.size() + ", Temp Items
Size: " + tempItems.size() + ", Energy Data List Size: " + energyDataList.size());
                for (EnergyData energyData : tempItems) {
                    if
(energyData.getActive_energy_in().toLowerCase().contains(constraint.toString().toLowe
rCase())                                                                    ||
energyData.getActive_power().toLowerCase().contains(constraint.toString().toLowerCase
())                                                                         ||
energyData.getCurrent().toLowerCase().contains(constraint.toString().toLowerCase())
                            ||
energyData.getVoltage().toLowerCase().contains(constraint.toString().toLowerCase())
||
energyData.getFrequency().toLowerCase().contains(constraint.toString().toLowerCase())
||
energyData.getPower_factor().toLowerCase().contains(constraint.toString().toLowerCase
())                                                                         ||
energyData.getUpdate_date().toLowerCase().contains(constraint.toString().toLowerCase(
)))                                                                         {
                        suggestions.add(energyData);
                    }
                }
                FilterResults filterResults = new FilterResults();
                filterResults.values = suggestions;
                filterResults.count = suggestions.size();
                return filterResults;
            } else {
                return new FilterResults();
            }
        }

        @Override
        protected void publishResults(CharSequence constraint, FilterResults results)
{
            List<EnergyData> filterList = (ArrayList<EnergyData>) results.values;
            if (results != null && results.count > 0) {
                energyDataList.clear();
                for (EnergyData energyData : filterList) {
                    //Log.e("Filter:","I am here");
```

```
                    energyDataList.add(energyData);
                    notifyDataSetChanged();
                }
            }
            else{
                energyDataList.clear();
                //Log.e("Filter:             ","I             am             here11");
                notifyDataSetChanged();
            }
        }
    };

    @Override
    public                  int                  getItemCount()                  {
        return                                          energyDataList.size();
    }
}
```