ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Δικτύου Αισθητήρων για την Καταγραφή και Διαχείριση Κατανάλωσης Ηλεκτρικής Ενέργειας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Βασιλείου Καλιμάντζαλη-Λιανά

Επιβλέπων Καθηγητής:

Ευάγγελος Β. Χριστοφόρου

Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Δικτύου Αισθητήρων για την Καταγραφή και Διαχείριση Κατανάλωσης Ηλεκτρικής Ενέργειας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Καλιμάντζαλη-Λιανά

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Μήνα Έτος.

Ευάγγελος Χριστοφόρου	Παναγιώτης Τσαραμπάρης	Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Μαρία Β. Καλιμάντζαλη-Λιανά

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της εργασίας ήταν η ανάπτυξη μιας συσκευής χαμηλού κόστους, μικρού μεγέθους και εύκολης στην χρήση, η οποία θα επιτρέπει την παρακολούθηση της ποιότητας της ηλεκτρικής ισχύος εξ αποστάσεως. Η εργασία χωρίζεται σε δύο κύρια μέρη: Το πρώτο είναι το θεωρητικό και αφορά σε όλα τα εργαλεία και αισθητήρες που χρησιμοποιήθηκαν, και το δεύτερο είναι το πειραματικό-κατασκευαστικό που περιλαμβάνει την περιγραφή της υλοποίησης για την καταγραφή παρακολούθηση των μετρήσεων και γενικότερα της ποιότητας της ενέργειας στην παρούσα εργασία.

Στο Κεφάλαιο 1, γίνεται μία σύντομη αναφορά στους λόγους και στην ανάγκη ελέγχου της ποιότητας της ηλεκτρικής ισχύος σήμερα, ώστε να αποφευχθούν προβλήματα όπως είναι η υπέρταση, η βύθιση τάσης, οι ταχείες διακυμάνσεις (φλίκερ) και οι παραμορφώσεις των αρμονικών συναρτήσεων τάσης.

Στο Κεφάλαιο 2, αναφέρονται όλα τα εργαλεία hardware που χρησιμοποιήθηκαν προκειμένου να γίνει η ανάπτυξη του προϊόντος όπως κάποιοι από τους υπάρχοντες αισθητήρες μέτρησης τάσης, ρεύματος και ισχύος, αλλά και μικροελεγκτών όπως Esp32 με μικρή αναφορά στα πλεονεκτήματα και τα μειονεκτήματα του καθενός. Επίσης, γίνεται περιγραφή software εργαλείων όπως την τοπική βάση δεδομένων την online βάση δεδομένων, το API προγραμματισμό σε C# όπως επίσης το User Interface (UI) για την γραφική αναπαράσταση των αποτελεσμάτων.

Το Κεφάλαιο 3, περιλαμβάνει το κατασκευαστικό τμήμα της συσκευής, καθώς και τα πειράματα που διεξήχθησαν με τη χρήση της και παρουσιάζει λεπτομερώς όλες τις μετρήσεις που έγιναν στο εργαστήριο, με τη βοήθεια ενός Esp32, του αισθητήρα έντασης ρεύματος ACS712 για 20A, του αισθητήρα τάσης ZMPT101B. Επιπρόσθετα, γίνεται σαφής ανάλυση όλων των βημάτων υλοποίησης του API υπεύθυνου για την επεξεργασία των μετρήσεων, της σύνδεσης του με είτε με την online Βάση Δεδομένων

είτε με την τοπική Βάση Δεδομένων και την αποστολή των μετρήσεων εκεί, σειριακά αλλά και μέσω Wfi των μετρήσεων από το Esp32 προς το API.

Λέξεις-κλειδιά: *API, Esp32, Wfi, Αισθητήρας Μέτρησης Έντασης Ρεύματος, Αισθητήρας Μέτρησης Τάσης*

Abstract

The purpose of this thesis was to develop a low cost device, small in size and easy to use, which will allow monitoring the quality of electricity remotely. The thesis is divided into two main parts: The first is the theoretical and concerns all the tools and sensors used, and the second is the experimental-construction that includes the description of the implementation to record the monitoring of measurements and in general the quality of energy in the present work.

In Chapter 1, a brief reference is made to the reasons and the need to control the quality of electrical power today, in order to avoid problems such as overshoot of voltage, voltage dropdown, rapid flickers and distortions of harmonic voltage functions.

Chapter 2 lists all the hardware tools used to develop the product, such as some of the existing voltage, current and power sensors, as well as microcontrollers such as Esp32 with a brief overview of the pros and cons. Software tools such as the local database, the online database, the C # programming API as well as the User Interface (UI) for graphical representation of the results are also described in depth.

Chapter 3, includes the components of the device, as well as the experiments performed using it and presents in detail all the measurements made in the laboratory, with the help of an Esp32, the ACS712 for 20A, the voltage sensor ZMPT101B. In addition, there is a clear analysis of all the steps of the implementation of the API responsible for the processing of the measurements, its connection with either the online Database or the local Database and the sending of the measurements there, serially but also through Wfi of the measurements from Esp32 to the API.

Keywords: *API, Esp32, Wfi, Current Measurement Sensor, Voltage Measurement Sensor*

Πρόλογος – Ευχαριστίες

Για αρχή, θα ήθελα να ευχαριστήσω τον καθηγητή Χριστοφόρου Ευάγγελο, της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, που μου ανέθεσε τη διπλωματική αυτή κατά τη διάρκεια του προπτυχιακού μου Προγράμματος "Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών". Η συμβολή του σε αυτή τη διπλωματική μέσω της καθοδήγησης και της επίβλεψης το ήταν σημαντική, τόσο με τη βιβλιογραφία που μου πρότεινε όσο και με το χρόνο που αφιέρωσε για την ολοκλήρωσή της.

Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Αγγελόπουλο Σπυρίδωνα για τη σημαντική επιστημονική βοήθεια που μου πρόσφερε, μέσω των προσωπικών του γνώσεων σχετικά με τους αισθητήρες και το Esp32, καθώς και για την καθοδήγηση που μου πρόσφερε στην κατασκευή και τη σύνδεση των μεμονωμένων αισθητήρων.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, τον Βασίλη και την Ειρήνη, και τον φίλο μου Ελευθέριο Τσόλκα για την υποστήριξη και μου παρέχουν όλο αυτό το διάστημα στα ακαδημαϊκά μου χρόνια

Περιεχόμενα

1 Εισαγωγή 20

1.1 Ποιότητα Ενέργειας 20

1.2 Περιγραφή της γενικής ροής της εφαρμογής 21

2 Θεωρητική Περιγραφή του Software και Hardware 22

2.1 ACS712 Αισθητήρας Μέτρησης Ρευματος 22

2.2 ZMPT101B Αισθητήρας Μέτρησης Τάσης 24

2.3 Επικοινωνία Api με Esp32 26

2.3.1 Αξιοπιστία του ESP32 27

2.3.2 Χαμηλή Κατανάλωση 27

2.3.3 Υψηλό Επίπεδο Ενσωμάτωσης 27

2.3.4 Wi-Fi και Bluetooth 28

2.3.5 Προδιαγραφές του ESP32 28

2.4 Γενική Ροή Δικτύων Πληροφορίας ή Δεδομένων 29

2.5 WiFi 30

2.6 Χρήσεις Wifi 30

2.6.1 Αρχές Λειτουργίας 31

2.6.2 Απόδοση 33

2.6.3 Μεταδιδόμενη Ισχύς 33

2.6.4 MIMO (multiple-input and multiple-output) 34

2.7 Μοντέλο Πελάτη - Διακομιστή (Client – Server Model) 35

2.7.1 Λειτουργία του Μοντέλου 35

2.7.2 Πρωτόκολλα Επικοινωνίας 36

2.7.3 Πρωτόκολλο User Datagram Protocol (UDP) 36

2.7.4 Αξιοπιστία 38

2.7.5 Πρωτόκολλο Transmission Control Protocol (TCP) 39

2.7.6 Αξιόπιστη Μετάδοση 40

2.8 Visual Studio 41

2.8.1 Τι είναι το Visual Studio 41

2.8.2 Γιατί Visual Studio 42

2.9 C# 42

2.9.1 Τί είναι C# 42

2.9.2 Βασικά χαρακτηριστικά αντικειμενοστραφή προγραμματισμού 43

2.9.3 Βασικά χαρακτηριστικά C# Γλώσσας Προγραμματισμού 43

2.9.4 C# - Μοντέρνα και εύκολη στην εκμάθηση 44

2.9.5 C# - Γρήγορη και ανοικτού κώδικα (open source) 45

2.9.6 C# - Πολλαπλών Πλατφορμών (Cross platform) 45

2.9.7 C# Ασφαλής και Αποτελεσματική 46

2.9.8 C# σαν πολυεργαλείο 46

2.10 User Interface (UI) 47

2.10.1 Windows Presentation Foundation (WPF) 47

2.10.2 Πλεονεκτήματα WPF 49

2.10.3 Τί είναι XAML 50

2.10.4 Σύνδεση Δεδομένων (data binding) μέσω WPF 50

2.10.5 Σύνδεση στη Βάση Δεδομένων 51

2.11 XAMPP 52

2.11.1 Apache 52

2.11.2 MySQL 52

2.11.3 PHP 53

2.11.4 Εφαρμογές του XAMPP 53

2.11.5 Πρακτική επεξήγηση χρήσης του XAMPP 54

2.12 Online DataBase 57

2.12.1 Χρήση λογισμικού όπως το MySQL Workbench για τη διαχείριση της διαδικτυακής βάσης δεδομένων 57

2.12.2 PHP 58

3 Ανάπτυξη του προϊόντος Energy Meter 59

3.1 Αισθητήρας Ρεύματος 59

3.2 Αισθητήρας Τάσης 60

3.2.1 Αποστολή δεδομένων από Esp32 προς το API 60

3.3 Ροή επικοινωνίας μεταξύ Esp32 και Api 61

3.3.1 Ρύθμιση Μεταβλητών (Configuration) του Esp32 για Wifi mode 62

3.3.2 API Configuration για Wifi mode 64

3.4 Ροή επικοινωνίας μεταξύ Esp32 και Api μέσω Serial Port 65

3.4.1 Esp32 Configuration για Serial Port mode 66

3.4.2 Api Configuration για Serial Port mode 66

3.5 PHP manager 71

3.6 Online DataBase 72

3.6.1 DataBase Query 73

3.7 Local DataBase 75

3.7.1 DataBaseManager 75

3.7.2 DataBase Structure 76

3.7.3 DataBase Query 78

3.8 User Interface 79

3.9 Υπολογισμοί στο API 83

3.9.1 Υπέρταση ή Πτώση Τάσης και υπολογισμοί 83

3.9.2 Συντελεστής Ισχύος Cp και Υπολογισμών 85

4 Συμπεράσματα 89

5 Βιβλιογραφία 90

6 Παράρτημα 92

1 Εισαγωγή

1.1 Ποιότητα Ενέργειας

Ο σκοπός αυτού του κεφαλαίου είναι να παρουσιάσει το μεγάλο πρόβλημα της μέτρησης της ποιότητας της ενέργειας. Έχει παρατηρηθεί ότι οι διατάξεις με τις οποίες τα δημόσια βιομηχανικά κτίρια και οι ιδιωτικοί οικισμοί επιδιώκουν την μέτρηση της ενέργειας περιλαμβάνουν σύγχρονα συστήματα αυτοματισμού κτιρίων, τα οποία αποτελούνται κυρίως από αισθητήρες, ελεγκτές και ενεργοποιητές προκειμένου να επιτύχουν την όσο μεγαλύτερη δυνατή απόδοση ενέργειας. Ωστόσο, αξίζει να σημειωθεί ότι η απόδοση της ενέργειας μετριέται κυρίως σε ποσοτικό επίπεδο (επιδιώκοντας την άμεση εξοικονόμηση) και λιγότερο σε ποιοτικό επίπεδο (επιδιώκοντας την έμμεση εξοικονόμηση)

Υπάρχουν πολλά οφέλη από την κατανάλωση υψηλής ποιότητας ενέργειας όπως αποφυγή ζημιών στον εξοπλισμό, μείωση του κόστους από διακοπή ρεύματος κ.λπ. Οι σημερινοί χρήστες ενέργειας γνωρίζοντας πλέον τα προαναφερθέντα οφέλη, επιδιώκουν την βελτίωση της ποιότητας ενέργειας, ωθώντας έτσι την παγκόσμια κοινότητα να αναπτύξει αισθητήρες και μικροεπεξεργαστές σε τέτοιο βαθμό, ώστε σε οποιαδήποτε πιθανή ενέργεια διακύμανση θα επιτευχθεί η χαμηλότερη δυνατή ευαισθησία.

Τα συστήματα ηλεκτρικής ενέργειας σε όλο τον κόσμο αντιμετωπίζουν ριζικό μετασχηματισμό με την ανάγκη να αποσυμφορηθούν από την παροχή ηλεκτρικής ενέργειας, να αντικατασταθούν τα γηράσκοντα περιουσιακά στοιχεία και να αξιοποιηθούν νέες τεχνολογίες πληροφοριών και επικοινωνιών (ΤΠΕ). Το Smart Grid χρησιμοποιεί προηγμένες ΤΠΕ για τον έλεγχο αξιόπιστων και αποτελεσματικών συστημάτων ισχύος επόμενης γενιάς.[1]

Για τους παραπάνω λόγους αναπτύχθηκε το εν λόγω προϊόν που αποτελεί ένα όργανο μέτρησης και παρακολούθησης της ποιότητας ενέργειας ως Smart Grid εφαρμογή και η παρούσα διπλωματική επεξηγεί και περιγράφει ενδελεχώς τη λειτουργία του.

1.2 Περιγραφή της γενικής ροής της εφαρμογής

Η διαδικασία υλοποίησης χωριζεται στα εξης μερη:

1. Το ηλεκτρολογικό κομμάτι που είναι η υλοποίηση του ηλεκτρικού κυκλώματος, τη σύνδεση της πρίζας με τον αισθητήρα ρεύματος και τάσης που στέλνουν τις μετρήσεις τους στον esp32.

2. Την αποστολή των μετρήσεων από τον esp32 προς το API είτε μέσω καλωδίου usb (serial port) είτε μέσω wifi.

3. Την επεξεργασία των δεδομένων μέσω του API σε C#. Το API είναι υπεύθυνο για την αποστολή των μετρήσεων σε

4. Το API είναι υπεύθυνο για την αποστολή των μετρήσεων στην βάση δεδομένων σε MySQL. Η βάση είναι αποθηκευμένη είτε τοπικά (στον ηλεκτρονικό υπολογιστή εκεί που τρέχει και το API) είτε online

5. Την επεξεργασία των δεδομένων μέσω του API (σε C#) την υλοποίηση των απαραίτητων υπολογισμών και την εξαγωγή των αντίστοιχων αποτελεσμάτων-συμπερασμάτων και την μεταφορά τους στο UI

6. Την υλοποίηση του UI μέσω XAML για εμφάνιση δεδομένων/αποτελεσμάτων στον υπολογιστή

Τα ακόλουθα κεφάλαια περιέχουν την εξήγηση αυτών των βημάτων.

2 Θεωρητική Περιγραφή του Software και Hardware

2.1 ACS712 Αισθητήρας Μέτρησης Ρευματος

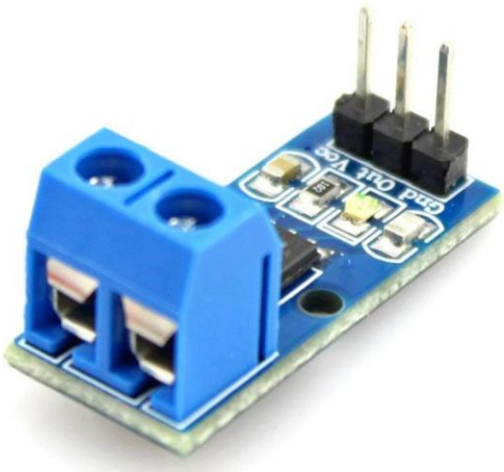
Η μονάδα ACS712, που εμφανίζεται στην εικόνα 2.1, χρησιμοποιεί το διάσημο IC ACS712 για τη μέτρηση του ρεύματος χρησιμοποιώντας την αρχή Hall Effect. Η μονάδα παίρνει το όνομά της από το IC (ACS712) που χρησιμοποιείται στη μονάδα.

Αυτή η μονάδα ACS712 μπορεί να μετρήσει το ρεύμα ρεύματος AC ή DC που κυμαίνεται από + 5A έως -5A, + 20A έως -20A και + 30A έως -30A. Στην περίπτωσή μας χρησιμοποιήθηκε το εύρος + 20A έως -20A Αυτή η μονάδα εξάγει αναλογική τάση (0-5V) με βάση το ρεύμα που ρέει μέσω του καλωδίου. Ως εκ τούτου, είναι πολύ εύκολο να διασυνδεθεί αυτός ο αισθητήρας με οποιονδήποτε μικροελεγκτή.

Το Allegro ACS715 παρέχει οικονομικά και ακριβή λύσεις για ανίχνευση ρεύματος DC σε συστήματα. Το πακέτο συσκευών επιτρέπει την εύκολη εφαρμογή από τον πελάτη. Οι τυπικές εφαρμογές περιλαμβάνουν έλεγχο κινητήρα, ανίχνευση φορτίου και διαχείριση, τροφοδοτικά εναλλαγής και υπερβολικό ρεύμα προστασία από σφάλματα.

Η συσκευή αποτελείται από ένα ακριβές, γραμμικό κύκλωμα Hall χαμηλής μετατόπισης με μια διαδρομή αγωγού χαλκού που βρίσκεται κοντά στην επιφάνεια του καλούπι. Εφαρμοσμένο ρεύμα που ρέει μέσω αυτής της αγωγιμότητας χαλκού. Η διαδρομή δημιουργεί ένα μαγνητικό πεδίο που το Hall Integrated Circuit μετατρέπει σε αναλογική τάση. Η ακρίβεια της συσκευής βελτιστοποιείται μέσω της εγγύτητας του μαγνητικού σήματος με τον μετατροπέα Hall. Μια ακριβής αναλογική τάση παρέχεται από το χαμηλό όφσετ, BiCMOS Hall IC σταθεροποιημένο με chopper, το οποίο είναι προγραμματισμένο για ακρίβεια μετά τη συσκευασία.

Η έξοδος της συσκευής έχει θετική κλίση (> VIOUT (Q)) όταν ένα αυξανόμενο ρεύμα ρέει μέσω της πρωτεύουσας διαδρομής αγωγού χαλκού (από τους πείρους 1 και 2, στις ακίδες 3 και 4), που είναι η διαδρομή που χρησιμοποιείται για την ανίχνευση ρεύματος. Η εσωτερική αντίσταση αυτής της αγώγιμης διαδρομής είναι τυπική 1,2 mΩ, παρέχοντας χαμηλή απώλεια ισχύος. Το πάχος του αγωγού χαλκού επιτρέπει την επιβίωση της συσκευής σε συνθήκες υπερέντασης έως 5×. Οι ακροδέκτες της αγώγιμης διαδρομής απομονώνονται ηλεκτρικά από τους αγωγούς IC του αισθητήρα (ακίδες 5 έως 8). Αυτό επιτρέπει στο IC του αισθητήρα ρεύματος ACS712 να χρησιμοποιείται σε εφαρμογές που απαιτούν ηλεκτρική απομόνωση χωρίς τη χρήση οπτομονωτικών ή άλλων δαπανηρών τεχνικών απομόνωσης.[2]



Σχήμα 2.1: ACS712 Αισθητήρας Ρεύματος[3].

2.2 ZMPT101B Αισθητήρας Μέτρησης Τάσης

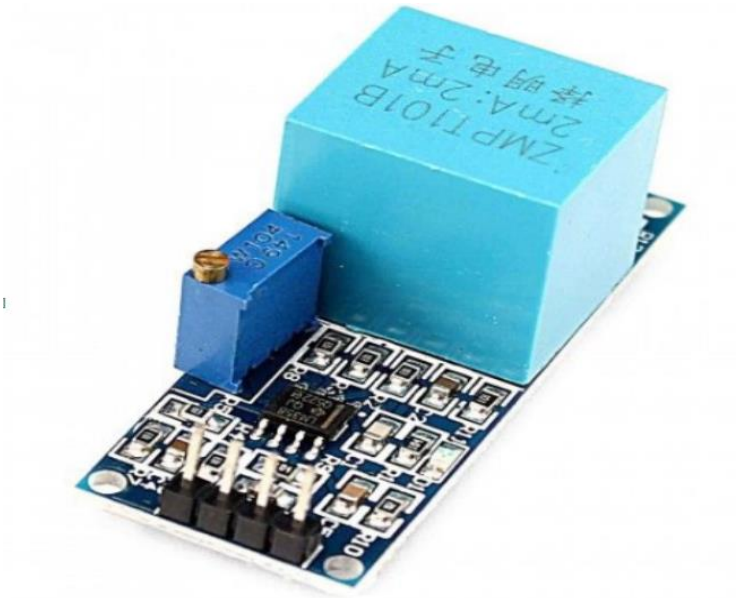
ZMPT101B είναι ένας μετασχηματιστής τάσης ιδανικός για τη μέτρηση της τάσης AC. Έχει υψηλή ακρίβεια, καλή συνέπεια για μέτρηση τάσης και ισχύος έως 250V AC. Εύκολο στη χρήση και διαθέτει ποτενσιόμετρο πολλαπλών στροφών για ρύθμιση της εξόδου ADC, όπως φαίνεται στην εικόνα 2.2.

Αυτός ο αισθητήρας χρησιμοποιήθηκε για τη μέτρηση της τάσης μιας συσκευής. Ο μετατροπέας AC είναι μονοφασικός και συνδέεται με καλώδιο στην πλακέτα Esp32. Η μέγιστη τάση καθορίζεται πάντα από την μέγιστη τάση AD του βρόχου όπου πραγματοποιείται η δειγματοληψία. Στο Esp32 αυτή η τάση AD είναι πάντα 3.3V. Ο αισθητήρας λειτουργεί κανονικά όταν υπάρχει ρεύμα μεταξύ 1-2 mA. Όταν η τάση εισόδου είναι <100V, τότε το ρεύμα λειτουργίας είναι ακριβώς στα 2mA, ενώ όταν είναι μεταξύ 100 και 220 V, τότε το ρεύμα ανήκει στην περιοχή 1 <I <2 mA.[4].

Για τάση μεγαλύτερη από 220V, η αντίσταση πρέπει να αλλάξει, διαφορετικά ο αισθητήρας πιθανότατα θα υποστεί ζημιά.

Το κανονικό εύρος θερμοκρασίας λειτουργίας του αισθητήρα είναι από -40 ° C έως + 60 ° C, η γραμμικότητα των φορτίων είναι <0,2%, ενώ η τάση απομόνωσης είναι περίπου 4000V.

Ο υπό εξέταση αισθητήρας φαίνεται στο Σχήμα 2:



Σχήμα 2.2: ZMPT101B Αισθητήρας Τάσης [5].

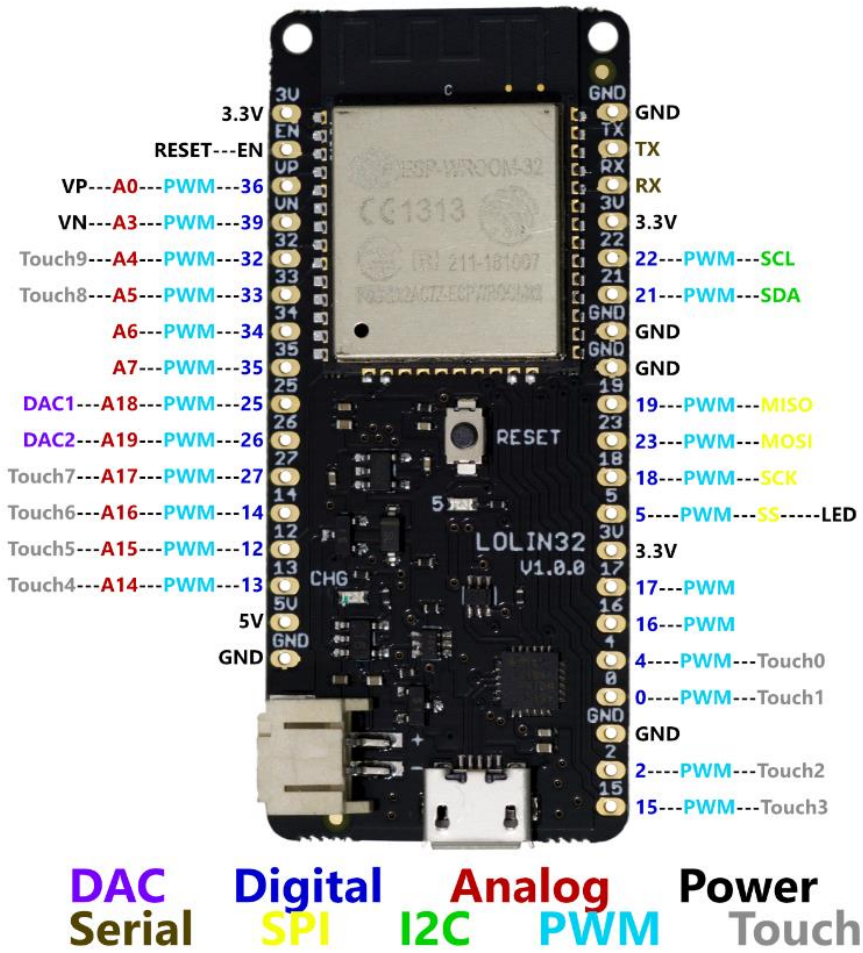
Για τη βελτίωση της αξιοπιστίας, η αντίσταση που επιλέγεται για τον περιορισμό του ρεύματος είναι τέσσερις φορές την ονομαστική τιμή και ο συντελεστής θερμοκρασίας είναι πάντα υψηλός. Κατά την επιλογή δειγματοληψίας, η αντίσταση δεν πρέπει να υπερβαίνει την τιμή που δίνεται στην σχέση (2.1):

$$\frac{Vout(max)}{Vin(max)}*\mathrm{R} \qquad (2.1)$$

Με την τιμή του R να είναι γνωστή από την αρχή από τον κατασκευαστή.

2.3 Επικοινωνία Api με Esp32

Το ESP32 είναι μια μονάδα μικροελεγκτή πλούσια σε χαρακτηριστικά (MCU) με ενσωματωμένη σύνδεση Wi-Fi και Bluetooth για ένα ευρύ φάσμα εφαρμογών. Είναι πολύ εύκολο να συνδεθεί ηλεκτρικά με αισθητήρες και να επεξεργαστεί τα δεδομένα που λαμβάνει από αυτούς και παρουσιάζεται στην εικόνα 2.3.



Σχήμα 2.3: Το μοντέλο του ESP32 που χρησιμοποιήθηκε[6].

2.3.1 Αξιοπιστία του ESP32

Το ESP32 μπορεί να λειτουργεί αξιόπιστα σε βιομηχανικά περιβάλλοντα, με θερμοκρασία λειτουργίας που κυμαίνεται από –40 ° C έως + 125 ° C. Με την υποστήριξη προηγμένων κυκλωμάτων βαθμονόμησης, το ESP32 μπορεί να αφαιρέσει δυναμικά τις ατέλειες εξωτερικού κυκλώματος και να προσαρμοστεί σε αλλαγές σε εξωτερικές συνθήκες.

2.3.2 Χαμηλή Κατανάλωση

Σχεδιασμένο για κινητές συσκευές, φορητά ηλεκτρονικά και εφαρμογές IoT, το ESP32 επιτυγχάνει εξαιρετικά χαμηλή κατανάλωση ενέργειας με συνδυασμό διαφόρων τύπων ιδιόκτητου λογισμικού. Το ESP32 περιλαμβάνει επίσης υπερσύγχρονες λειτουργίες, όπως λεπτή πύλη ρολογιού, διάφορες λειτουργίες ισχύος και δυναμική κλιμάκωση ισχύος.

2.3.3 Υψηλό Επίπεδο Ενσωμάτωσης

Το ESP32 είναι εξαιρετικά εύκολο στην ενσωμάτωση και προσαρμογή σε διάφορα κυκλώματα με ενσωματωμένους διακόπτες κεραίας, RF balun, ενισχυτή ισχύος, ενισχυτή χαμηλού θορύβου, φίλτρα και μονάδες διαχείρισης ισχύος. Το ESP32 προσθέτει ανεκτίμητη λειτουργικότητα και ευελιξία στις εφαρμογές σας με ελάχιστες απαιτήσεις Printed Circuit Board (PCB).

2.3.4 Wi-Fi και Bluetooth

Το ESP32 μπορεί να λειτουργήσει ως πλήρες αυτόνομο σύστημα ή ως υποτελής συσκευή σε ένα κεντρικό MCU, μειώνοντας την εναέρια στοίβα επικοινωνίας στον κύριο επεξεργαστή εφαρμογών. Το ESP32 μπορεί να διασυνδεθεί με άλλα συστήματα για να παρέχει λειτουργικότητα Wi-Fi και Bluetooth μέσω των διεπαφών SPI / SDIO ή I2C / UART. [7]

2.3.5 Προδιαγραφές του ESP32

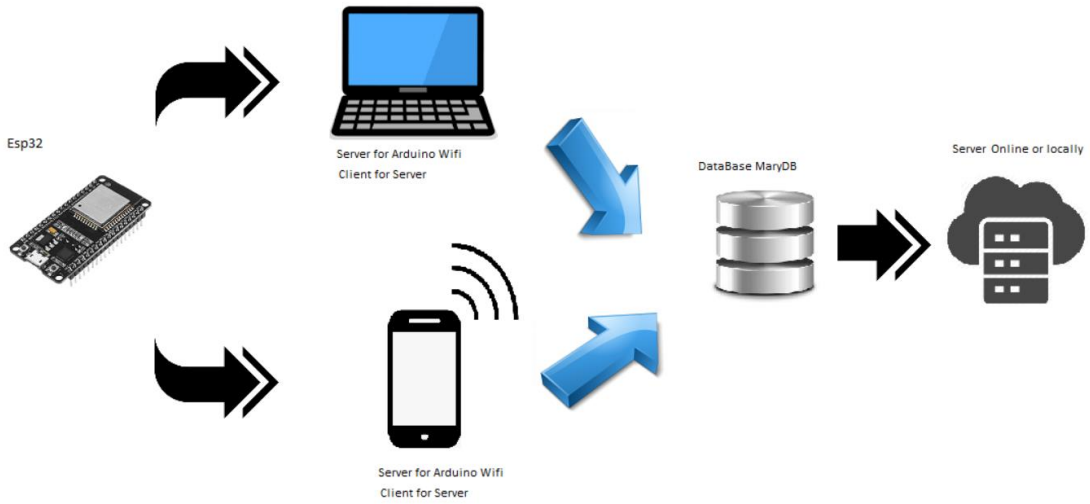
Οι προδιαγραφές του ESP32 που χρησιμοποιήθηκαν είναι:

- Υποστηριζόμενο πρωτόκολλο: 802.11 b/g/n
- Συχνότητα: 2.4 GHz με 2.5 GHz
- Τάση λειτουργίας: 5 V
- Τάση λειτουργίας των πιν μεταφοράς δεδομένων: 3.3 V
- Product Type: WiFi Development Tools
- Τρόπος σύνδεσης:
    - Σειριακά (USB micro)
    - Μέσω πίν (pin strips)

Για τους σκοπούς της παρούσας διπλωματικής, δημιουργήσαμε μια επικοινωνία WiFi μεταξύ του Api και του Esp32 για τη μετάδοση δεδομένων. Υποστηρίζεται επίσης σειριακή επικοινωνία μέσω καλωδίου.

2.4 Γενική Ροή Δικτύων Πληροφορίας ή Δεδομένων

Προκειμένου να επιτευχθεί μια γενική κατανόηση της υλοποίησης των δικτύων, παρέχεται ένας σχηματικός σχεδιασμός.



Σχήμα 2.4: Γενική Ροή Δικτύων.

Το πρώτο στάδιο είναι η αποστολή δεδομένων από το Esp32 σε μια υπολογιστική συσκευή μέσω wifi ή μέσω σειριακού καλωδίου (USB). Αυτή η υπολογιστική συσκευή θα μπορούσε να είναι φορητός υπολογιστής, επιτραπέζιος υπολογιστής, κινητό τηλέφωνο, tablet κ.λπ. Στην τρέχουσα διπλωματική, η υπολογιστική συσκευή που χρησιμοποιείται είναι φορητός ή επιτραπέζιος υπολογιστής. Το δεύτερο στάδιο είναι η υπολογιστική συσκευή να στέλνει δεδομένα στη βάση δεδομένων. Η βάση δεδομένων πρέπει να αποθηκευτεί σε διακομιστή (server). Αυτός ο διακομιστής θα μπορούσε να βρίσκεται στη συσκευή υπολογιστών που σημαίνει ότι θα ήταν προσβάσιμος μόνο από τη υπολογιστική συσκευή ή θα μπορούσε να αποθηκευτεί στο διαδίκτυο (online), το οποίο θα είναι προσβάσιμο από οποιονδήποτε έχει τα διαπιστευτήρια για πρόσβαση στη βάση δεδομένων. Όλα αυτά τα στάδια εξηγούνται σε βάθος στα επόμενα κεφάλαια.

2.5 WiFi

Αυτό το κεφάλαιο επικεντρώνεται στον τρόπο μετάδοσης δεδομένων από Esp32 στον υπολογιστή μέσω Wifi. Είναι συνετό να ξεκινήσει κάποιος με τη θεωρητική εξήγηση των βασικών αρχών λειτουργίας του Wi-Fi. Το Wi-Fi είναι μια οικογένεια πρωτοκόλλων ασύρματου δικτύου, με βάση την οικογένεια προτύπων IEEE 802.11, τα οποία χρησιμοποιούνται συνήθως για τοπική δικτύωση συσκευών και πρόσβαση στο Διαδίκτυο.

2.6 Χρήσεις Wifi

Η τεχνολογία Wi-Fi μπορεί να χρησιμοποιηθεί για την παροχή τοπικού δικτύου (local network) και πρόσβασης στο Διαδίκτυο σε συσκευές που βρίσκονται εντός του εύρους Wi-Fi ενός ή περισσοτέρων δρομολογητών που είναι συνδεδεμένοι στο Διαδίκτυο. Η κάλυψη ενός ή περισσοτέρων διασυνδεδεμένων σημείων πρόσβασης (interconnected access points-hotspots) μπορεί να εκτείνεται από μια περιοχή τόσο μικρή όσο λίγα δωμάτια έως και μερικά τετραγωνικά χιλιόμετρα. Η κάλυψη στην ευρύτερη περιοχή ενδέχεται να απαιτεί ομάδα σημείων πρόσβασης (access points) με αλληλεπικαλυπτόμενη κάλυψη.

Το Wi-Fi παρέχει υπηρεσίες σε ιδιωτικές κατοικίες, επιχειρήσεις, καθώς και σε δημόσιους χώρους. Τα hotspot Wi-Fi μπορούν να ρυθμιστούν είτε δωρεάν είτε εμπορικά, συχνά χρησιμοποιώντας μια ιστοσελίδα δεσμευμένης πύλης για πρόσβαση (captive portal webpage for access). Οργανισμοί, αρχές και επιχειρήσεις, όπως αεροδρόμια, ξενοδοχεία και εστιατόρια, συχνά παρέχουν δωρεάν ή επί πληρωμή hotspots για να προσελκύσουν πελάτες, να παρέχουν υπηρεσίες για την προώθηση επιχειρήσεων σε επιλεγμένες περιοχές.[8]

2.6.1 Αρχές Λειτουργίας

Οι σταθμοί Wi-Fi επικοινωνούν ανταλλάσσοντας πακέτα δεδομένων: μπλοκ δεδομένων που αποστέλλονται και παραδίδονται μεμονωμένα μέσω ραδιοκυμάτων. Όπως συμβαίνει με όλα τα ραδιοκύματα, αυτό γίνεται με τη διαμόρφωση και την αποδιαμόρφωση των κυμάτων μεταφοράς. Διαφορετικές εκδόσεις του Wi-Fi χρησιμοποιούν διαφορετικές τεχνικές, το 802.11b χρησιμοποιεί DSSS σε έναν μόνο φορέα, ενώ τα 802.11a, Wi-Fi 4, 5 και 6 χρησιμοποιούν πολλούς φορείς σε ελαφρώς διαφορετικές συχνότητες εντός του καναλιού.

Το πρότυπο 802.11 παρέχει αρκετά διαφορετικά εύρη ραδιοσυχνοτήτων για χρήση σε επικοινωνίες Wi-Fi: ζώνες 900 MHz, 2,4 GHz, 5 GHz, 5,9 GHz και 60 GHz. Κάθε εύρος χωρίζεται σε ένα πλήθος καναλιών. Οι χώρες εφαρμόζουν τους δικούς τους κανονισμούς στα επιτρεπόμενα κανάλια, τους επιτρεπόμενους χρήστες και τα μέγιστα επίπεδα ισχύος σε αυτά τα εύρη συχνοτήτων. Τα εύρη ζώνης ISM χρησιμοποιούνται επίσης συχνά.

Καθώς οι προδιαγραφές 802.11 εξελίχθηκαν για να υποστηρίξουν υψηλότερη απόδοση, τα πρωτόκολλα έχουν γίνει πολύ πιο αποτελεσματικά στη χρήση του εύρους ζώνης. Επιπλέον, έχουν αποκτήσει τη δυνατότητα να συγκεντρώσουν κανάλια μαζί για να αποκτήσουν ακόμη περισσότερη απόδοση όπου το εύρος ζώνης είναι διαθέσιμο. Το 802.11n επιτρέπει διπλό ραδιοφάσμα / εύρος ζώνης (κανάλια 40 MHz- 8) σε σύγκριση με 802.11a ή 802.11g (20 MHz). Το 802.11n μπορεί επίσης να ρυθμιστεί ώστε να περιορίζεται σε εύρος ζώνης 20 MHz για την αποφυγή παρεμβολών σε πυκνές κοινότητες. [9] Στη ζώνη των 5 GHz, επιτρέπονται σήματα εύρους ζώνης 20 MHz, 40 MHz, 80 MHz και 160 MHz με ορισμένους περιορισμούς, δίνοντας πολύ πιο γρήγορες συνδέσεις.

Το Wi-Fi ανήκει στην οικογένεια πρωτοκόλλων IEEE 802. Τα δεδομένα οργανώνονται σε 802.11 πλαίσια που μοιάζουν πολύ με τα πλαίσια Ethernet στο επίπεδο σύνδεσης δεδομένων, αλλά με επιπλέον πεδία διευθύνσεων. Οι διευθύνσεις MAC χρησιμοποιούνται ως διευθύνσεις δικτύου για δρομολόγηση μέσω LAN.

Δημιουργούνται και συντηρούνται από την επιτροπή προτύπων IEEE LAN / MAN (IEEE 802). Η βασική έκδοση του προτύπου κυκλοφόρησε το 1997 και είχε πολλές επακόλουθες τροποποιήσεις. Το πρότυπο και οι τροποποιήσεις παρέχουν τη βάση για προϊόντα ασύρματου δικτύου που χρησιμοποιούν τη μάρκα Wi-Fi. Ενώ κάθε τροποποίηση ανακαλείται επισήμως όταν ενσωματώνεται στην τελευταία έκδοση του προτύπου, ο εταιρικός κόσμος τείνει να εμπορεύεται τις αναθεωρήσεις επειδή υποδηλώνει συνοπτικά τις δυνατότητες των προϊόντων τους. [9] Ως αποτέλεσμα, στην αγορά, κάθε αναθεώρηση τείνει να γίνει το δικό της πρότυπο.

Εκτός από το 802.11, η οικογένεια πρωτοκόλλων IEEE 802 έχει συγκεκριμένες διατάξεις για Wi-Fi. Αυτά απαιτούνται επειδή τα μέσα που βασίζονται σε καλώδια Ethernet δεν κοινοποιούνται συνήθως, ενώ με ασύρματα όλες οι μεταδόσεις λαμβάνονται από όλους τους σταθμούς εντός του εύρους που χρησιμοποιούν αυτό το ραδιοφωνικό κανάλι. Ενώ το Ethernet έχει ουσιαστικά αμελητέα ποσοστά σφάλματος, τα ασύρματα μέσα επικοινωνίας υπόκεινται σε σημαντικές παρεμβολές. Επομένως, η ακριβής μετάδοση δεν είναι εγγυημένη. Εξαιτίας αυτού, για το Wi-Fi, το Logical Link Control (LLC) που καθορίζεται από το IEEE 802.2 χρησιμοποιεί πρωτόκολλα ελέγχου πρόσβασης πολυμέσων (MAC) του Wi-Fi για τη διαχείριση των επαναλήψεων χωρίς να βασίζεται σε υψηλότερα επίπεδα της στοίβας πρωτοκόλλων.

2.6.2 Απόδοση

Το εύρος λειτουργίας του Wi-Fi εξαρτάται από παράγοντες όπως η ζώνη συχνοτήτων, η έξοδος της ισχύος των ραδιοκυμάτων, η ευαισθησία του δέκτη, το κέρδος της κεραίας και ο τύπος της κεραίας, καθώς και η τεχνική διαμόρφωσης.

Σε μεγαλύτερες αποστάσεις και με μεγαλύτερη απορρόφηση σήματος ισχύος, η ταχύτητα συνήθως μειώνεται.

2.6.3 Μεταδιδόμενη Ισχύς

Σε σύγκριση με τα κινητά τηλέφωνα και παρόμοια τεχνολογία, οι πομποί Wi-Fi είναι συσκευές χαμηλής ισχύος. Γενικά, η μέγιστη ισχύς που μπορεί να μεταδώσει μια συσκευή Wi-Fi περιορίζεται από τους τοπικούς κανονισμούς, όπως το FCC Part 15 στις ΗΠΑ. Η ισοδύναμη ισοτροπικά ακτινοβολούμενη ισχύς (EIRP) στην Ευρωπαϊκή Ένωση περιορίζεται στα 20 dBm (100 mW).

Για την κάλυψη απαιτήσεων για εφαρμογές ασύρματου LAN, το Wi-Fi έχει μεγαλύτερη κατανάλωση ενέργειας σε σύγκριση με ορισμένα άλλα πρότυπα που έχουν σχεδιαστεί για την υποστήριξη εφαρμογών ασύρματου προσωπικού δικτύου (PAN). Για παράδειγμα, το Bluetooth παρέχει πολύ μικρότερο εύρος διάδοσης μεταξύ 1 και 100m και έτσι γενικά έχουν χαμηλότερη κατανάλωση ενέργειας. Άλλες τεχνολογίες χαμηλής ισχύος, όπως το ZigBee έχουν αρκετά μεγάλο εύρος, αλλά πολύ χαμηλότερο ρυθμό δεδομένων.

2.6.4 MIMO (multiple-input and multiple-output)

Το Wi-Fi 4 και τα υψηλότερα πρότυπα επιτρέπουν στις συσκευές να έχουν πολλαπλές κεραίες σε πομπούς και δέκτες. Οι πολλαπλές κεραίες επιτρέπουν στον εξοπλισμό να εκμεταλλεύεται τη διάδοση πολλαπλών διαδρομών στις ίδιες ζώνες συχνοτήτων παρέχοντας πολύ πιο υψηλότερες ταχύτητες και μεγαλύτερη εμβέλεια.Το Wi-Fi 4 μπορεί να υπερδιπλασιάσει την εμβέλεια σε σχέση με τα προηγούμενα πρότυπα.

Το πρότυπο Wi-Fi 5 χρησιμοποιεί αποκλειστικά τη ζώνη των 5 GHz και είναι ικανή για πολλαπλή σταθμική απόδοση WLAN τουλάχιστον 1 gigabit ανά δευτερόλεπτο, και μεμονωμένη απόδοση σταθμού τουλάχιστον 500 Mbit/s. Από το πρώτο τρίμηνο του 2016, η Wi-Fi Alliance πιστοποιεί συσκευές συμβατές με το πρότυπο 802.11ac ως "Wi-Fi CERTIFIED ac". Αυτό το πρότυπο χρησιμοποιεί αρκετές τεχνικές επεξεργασίας σήματος, όπως MIMO πολλαπλών χρηστών και 4X4 Spatial Multiplexing streams, και εύρος ζώνης των καναλιών (160 MHz) για την επίτευξη της απόδοσης gigabit. Σύμφωνα με μια μελέτη της IHS Technology, το 70% όλων των εσόδων από τις πωλήσεις σημείων πρόσβασης το πρώτο τρίμηνο του 2016 προήλθε από συσκευές 802.11ac.

2.7 Μοντέλο Πελάτη - Διακομιστή (Client – Server Model)

Το μοντέλο πελάτη-διακομιστή (Client–server model) είναι μια κατανεμημένης εφαρμογής δομή που χωρίζει εργασίες ή φόρτους εργασίας μεταξύ των παρόχων ενός πόρου ή υπηρεσίας, που ονομάζονται διακομιστές και αιτούντες υπηρεσίες, που ονομάζονται πελάτες. Συχνά οι πελάτες και οι διακομιστές επικοινωνούν μέσω δικτύου υπολογιστών σε ξεχωριστό υλικό(hardware), αλλά τόσο ο πελάτης όσο και ο διακομιστής ενδέχεται να βρίσκονται στο ίδιο σύστημα. Ένας κεντρικός υπολογιστής διακομιστή εκτελεί ένα ή περισσότερα προγράμματα διακομιστή, τα οποία μοιράζονται τους πόρους τους με πελάτες. Ένας πελάτης δεν μοιράζεται κανέναν από τους πόρους του, αλλά ζητά περιεχόμενο ή υπηρεσία από έναν διακομιστή. Οι πελάτες, επομένως, ξεκινούν συνεδρίες επικοινωνίας με διακομιστές, οι οποίοι περιμένουν εισερχόμενα αιτήματα. Παραδείγματα εφαρμογών υπολογιστών που χρησιμοποιούν το μοντέλο πελάτη-διακομιστή είναι το email, η εκτύπωση δικτύου και το World Wide Web. Για τους σκοπούς της παρούσας διπλωματικής εφαρμόσαμε αυτό το μοντέλο μεταξύ του Api και του Esp32.

2.7.1 Λειτουργία του Μοντέλου

Το χαρακτηριστικό πελάτη-διακομιστή περιγράφει τη σχέση των συνεργαζόμενων προγραμμάτων σε μια εφαρμογή. Ο πόρος του διακομιστή παρέχει μια λειτουργία ή υπηρεσία σε έναν ή πολλούς πελάτες, οι οποίοι ξεκινούν αιτήματα για τέτοιες υπηρεσίες. Οι διακομιστές ταξινομούνται βάσει των υπηρεσιών που παρέχουν. Για παράδειγμα, ένας διακομιστής ιστού εξυπηρετεί ιστοσελίδες και ένας διακομιστής αρχείων εξυπηρετεί αρχεία υπολογιστή. Ένας κοινόχρηστος πόρος μπορεί να είναι οποιοδήποτε από τα λογισμικά και τα ηλεκτρονικά στοιχεία του υπολογιστή του διακομιστή, από προγράμματα και δεδομένα έως επεξεργαστές και συσκευές αποθήκευσης. Η κοινή χρήση πόρων ενός διακομιστή αποτελεί μία υπηρεσία.

Το εάν ένας υπολογιστής είναι πελάτης, διακομιστής ή και τα δύο, καθορίζεται από τη φύση της εφαρμογής που απαιτεί τις λειτουργίες της υπηρεσίας. Για παράδειγμα, ένας μεμονωμένος υπολογιστής μπορεί να εκτελεί ταυτόχρονα διακομιστές ιστού και λογισμικό διακομιστή αρχείων για την εξυπηρέτηση διαφορετικών δεδομένων σε πελάτες που κάνουν διαφορετικά είδη αιτημάτων. Το λογισμικό πελάτη μπορεί επίσης να επικοινωνεί με λογισμικό διακομιστή στον ίδιο υπολογιστή. Η επικοινωνία μεταξύ διακομιστών, όπως ο συγχρονισμός δεδομένων, καλείται μερικές φορές επικοινωνία μεταξύ διακομιστών ή διακομιστών προς διακομιστή.

2.7.2 Πρωτόκολλα Επικοινωνίας

Υπάρχουν πολλά πρωτόκολλα που χρησιμοποιούνται για την επικοινωνία και τη μετάδοση δεδομένων για το μοντέλο διακομιστή-πελάτη. Δύο από τα βασικότερα αυτά πρωτόκολλα είναι τα TCP και UDP.

2.7.3 Πρωτόκολλο User Datagram Protocol (UDP)

Στα δίκτυα υπολογιστών, το User Datagram Protocol (UDP) είναι ένα από τα βασικά μέλη του συνόλου πρωτοκόλλων Διαδικτύου. Το πρωτόκολλο σχεδιάστηκε από τον David P. Reed το 1980. Με το UDP, οι εφαρμογές υπολογιστών μπορούν να στέλνουν μηνύματα, στην περίπτωση που αναφέρονται ως datagrams, σε άλλους κεντρικούς υπολογιστές(hosts) σε ένα Internet Protocol (IP) δίκτυο. Δεν απαιτείται προηγούμενη επικοινωνία για τη ρύθμιση καναλιών επικοινωνίας ή μονοπατιών (paths) δεδομένων.

Το UDP χρησιμοποιεί ένα απλό μοντέλο επικοινωνίας χωρίς σύνδεση με ελάχιστους μηχανισμούς πρωτοκόλλου. Το UDP παρέχει αρκετούς ελέγχους για διασφάλιση της ακεραιότητας των δεδομένων και αριθμούς θύρας για την αντιμετώπιση διαφορετικών λειτουργιών στην πηγή και τον προορισμό του datagram. Δεν έχει διαλόγους χειραψίας (handshaking dialogues) και έτσι εκθέτει το πρόγραμμα του χρήστη σε οποιαδήποτε αξιοπιστία του υποκείμενου δικτύου. Δεν υπάρχει εγγύηση παράδοσης, παραγγελίας ή διπλής προστασίας. Εάν απαιτούνται λειτουργίες διόρθωσης σφαλμάτων σε επίπεδο διασύνδεσης δικτύου, μια εφαρμογή μπορεί να χρησιμοποιήσει Transmission Control Protocol (TCP) ή Stream Control Transmission Protocol (SCTP) που έχουν σχεδιαστεί για το σκοπό αυτό.

Το UDP είναι κατάλληλο για σκοπούς όπου ο έλεγχος και η διόρθωση σφαλμάτων είτε δεν είναι απαραίτητοι είτε εκτελούνται στην εφαρμογή. Το UDP αποφεύγει τα γενικά έξοδα μιας τέτοιας επεξεργασίας στη στοίβα πρωτοκόλλων. Οι εφαρμογές που είναι ευαίσθητες στο χρόνο χρησιμοποιούν συχνά UDP επειδή είναι προτιμότερη η αποχώρηση πακέτων από την αναμονή καθυστέρησης πακέτων λόγω αναμετάδοσης, η οποία μπορεί να μην είναι επιλογή σε σύστημα πραγματικού χρόνου.[10]

Το UDP είναι κατάλληλο για σκοπούς όπου ο έλεγχος και η διόρθωση σφαλμάτων είτε δεν είναι απαραίτητοι είτε εκτελούνται στην εφαρμογή. Το UDP αποφεύγει τα γενικά έξοδα μιας τέτοιας επεξεργασίας στη στοίβα πρωτοκόλλων. Οι εφαρμογές που είναι ευαίσθητες στο χρόνο, χρησιμοποιούν συχνά UDP, επειδή είναι προτιμότερο να συνεχίσουν την λειτουργία παρά τις απώλειες που μπορεί να έχουν παρά να σταματήσουν την αναμετάδοση των πακέτων λόγω απωλειών πληροφορίας.

2.7.4 Αξιοπιστία

Χάρη στην αξιοπιστία, οι εφαρμογές UDP πρέπει να είναι πρόθυμες να δεχτούν κάποια απώλεια πακέτου, αναδιάταξη, σφάλματα ή επανάληψη. Εάν χρησιμοποιηθεί το UDP, οι εφαρμογές του τελικού χρήστη πρέπει να παρέχουν οποιαδήποτε απαραίτητη χειραψία, όπως επιβεβαίωση σε πραγματικό χρόνο ότι το μήνυμα έχει ληφθεί. Εάν μια εφαρμογή απαιτεί υψηλό βαθμό αξιοπιστίας, μπορεί να χρησιμοποιηθεί ένα πρωτόκολλο όπως το πρωτόκολλο ελέγχου μετάδοσης που όμως ξεπερνά τα όρια ενδιαφέροντος της παρούσας διπλωματικής.

Τις περισσότερες φορές, οι εφαρμογές UDP δεν χρησιμοποιούν μηχανισμούς αξιοπιστίας και μπορεί ακόμη και να παρεμποδίζονται από αυτούς. Τα μέσα ροής, τα παιχνίδια για πολλούς παίκτες σε πραγματικό χρόνο και η φωνή μέσω IP (VoIP) είναι παραδείγματα εφαρμογών που χρησιμοποιούν συχνά UDP. Σε αυτές τις συγκεκριμένες εφαρμογές, η απώλεια πακέτων δεν είναι συνήθως θανατηφόρο πρόβλημα. Στο VoIP, για παράδειγμα, ο λανθάνων χρόνος και το jitter είναι οι κύριες ανησυχίες. Η χρήση του TCP θα προκαλούσε jitter εάν χάνονταν πακέτα καθώς το TCP δεν παρέχει επακόλουθα δεδομένα στην εφαρμογή ενώ ζητά την αποστολή των δεδομένων που λείπουν.

Στα ηλεκτρονικά και τις τηλεπικοινωνίες, το jitter είναι η απόκλιση από την πραγματική περιοδικότητα ενός πιθανώς περιοδικού σήματος, συχνά σε σχέση με ένα σήμα ρολογιού αναφοράς.

Ωστόσο , λοιπόν και στην εν λόγω εφαρμογή δεν έγκειται σε κάποια από τις παραπάνω κατηγορίες η απώλεια κάποιου πακέτου μετρήσεων δεν κρίνεται επιθυμητή για αυτο και δεν έγινε επιλογή του συγκεκριμένου πρωτοκόλλου. Αντιθέτως επιλέχθηκε το TCP το οποίο επεξηγείται παρακάτω.

2.7.5 Πρωτόκολλο Transmission Control Protocol (TCP)

Το πρωτόκολλο ελέγχου μετάδοσης (TCP) είναι ένα από τα κύρια πρωτόκολλα Διαδικτύου. Προήλθε από την αρχική εφαρμογή του δικτύου στην οποία συμπλήρωσε το Πρωτόκολλο Διαδικτύου (IP). Επομένως, αναφέρεται συνήθως ως TCP / IP. Το TCP παρέχει αξιόπιστη, ταξινομημένη και ελεγχόμενη από σφάλμα παράδοση ροής οκτάδων (byte) μεταξύ εφαρμογών που εκτελούνται σε κεντρικούς υπολογιστές που επικοινωνούν μέσω δικτύου IP. Σημαντικές εφαρμογές Διαδικτύου, όπως το World Wide Web, το ηλεκτρονικό ταχυδρομείο, η απομακρυσμένη διαχείριση και η μεταφορά αρχείων βασίζονται στο TCP, το οποίο αποτελεί μέρος του επιπέδου μεταφοράς του TCP / IP.

Το TCP είναι προσανατολισμένο στη σύνδεση δηλαδή δημιουργείται σύνδεση μεταξύ πελάτη και διακομιστή πριν από την αποστολή δεδομένων. Ο διακομιστής πρέπει να “ακούει” (παθητικά ανοιχτός) για αιτήματα σύνδεσης από πελάτες πριν από τη σύνδεση. Αυτού του τύπου η σύνδεση, η αναμετάδοση και η ανίχνευση σφαλμάτων αυξάνουν την αξιοπιστία αλλά επιμηκύνουν την καθυστέρηση. Οι εφαρμογές που δεν απαιτούν αξιόπιστη υπηρεσία ροής δεδομένων ενδέχεται να χρησιμοποιούν το Πρωτόκολλο Datagram χρήστη (UDP), το οποίο παρέχει μια υπηρεσία αναμετάδοσης δεδομένων χωρίς σύνδεση που δίνει προτεραιότητα στον χρόνο έναντι της αξιοπιστίας. Το TCP χρησιμοποιεί αποφυγή συμφόρησης δικτύου. Ωστόσο, υπάρχουν ευπάθειες στο TCP, όπως άρνηση υπηρεσίας, παραβίαση σύνδεσης, βέτο TCP και επαναφορά επίθεσης. [10]

2.7.6 Αξιόπιστη Μετάδοση

Το TCP χρησιμοποιεί μία ακολουθία αριθμών για να προσδιορίσει κάθε byte δεδομένων. Η ακολουθία αριθμών προσδιορίζει τη σειρά των byte που αποστέλλονται από κάθε υπολογιστή, έτσι ώστε τα δεδομένα να μπορούν να ανακατασκευαστουν με τη σειρά, ανεξάρτητα από οποιαδήποτε αναδιάταξη πακέτων ή απώλεια πακέτου που μπορεί να συμβεί κατά τη μετάδοση.Η ακολουθία αριθμών του πρώτου byte επιλέγεται από τον πομπό για το πρώτο πακέτο, το οποίο φέρει την ένδειξη SYN. Αυτός ο αριθμός μπορεί να είναι αυθαίρετος και θα πρέπει, στην πραγματικότητα, να είναι απρόβλεπτος για την άμυνα ενάντια σε επιθέσεις πρόβλεψης ακολουθίας TCP.

Οι επικυρώσεις (Acknowledgements) “ACK” αποστέλλονται με μία νέα ακολουθία αριθμών από τον παραλήπτη δεδομένων για να ενημερώσουν τον αποστολέα ότι τα δεδομένα έχουν ληφθεί στο καθορισμένο byte. Οι ACK δεν υπονοούν ότι τα δεδομένα έχουν παραδοθεί στην εφαρμογή. Απλώς δηλώνουν ότι είναι πλέον ευθύνη του παραλήπτη να παραδώσει τα δεδομένα.

Η αξιοπιστία επιτυγχάνεται μέσω του αποστολέα ο οποίος είναι υπεύθυνος για τον εντοπισμό των χαμένων δεδομένων και την αναμετάδοση τους. Ενημερωτικά, το TCP χρησιμοποιεί δύο βασικές τεχνικές για τον εντοπισμό της απώλειας. Χρονικό όριο αναμετάδοσης (συντομευμένο ως RTO) και διπλές αθροιστικές επιβεβαιώσεις (DupAcks). Ωστόσο δεν κρίνεται απαραίτητο να εξηγηθούν περεταίρω οι προαναφερθέντες τεχνικές.

Εν κατακλείδι, για την επικοινωνία μεταξύ του Api και του Esp32 βασίστηκε στη δυνατότητα WiFi του esp32 για τη δημιουργία σύνδεσης. Εκτός από τη μετάδοση δεδομένων μεταξύ τους (του Api και του Esp32), χρησιμοποιήθηκε το μοντέλο διακομιστή πελάτη και το πρωτόκολλο TCP για σταθερότητα και εγγύηση επικύρωσης των δεδομένων.

2.8 Visual Studio

2.8.1 Τι είναι το Visual Studio

Το εργαλείο που χρησιμοποιείται για την ανάπτυξη του κώδικα λογισμικού σε C# (υπεύθυνο για την επεξεργασία των μετρήσεων, την εκτέλεση των απαραίτητων υπολογισμών, την αποστολή τους στη βάση δεδομένων και την εμφάνιση τους σε περιβάλλον γραφικών διεπαφής χρήστη στον χρήστη) είναι το Visual Studio.

Το Visual Studio είναι ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE) που αναπτύχθηκε από τη Microsoft για την ανάπτυξη GUI (Γραφικό περιβάλλον χρήστη), κονσόλας, εφαρμογών Ιστού, εφαρμογών ιστού, εφαρμογών για κινητά, cloud και διαδικτυακών υπηρεσιών κ.λπ. Με τη βοήθεια αυτού του IDE, μπορεί δημιουργήσει διαχειριζόμενο κώδικα καθώς και εγγενή κώδικα. Χρησιμοποιεί τις διάφορες πλατφόρμες λογισμικού ανάπτυξης λογισμικού της Microsoft, όπως το κατάστημα Windows, το Microsoft Silverlight και το API των Windows κ.λπ. Δεν είναι ειδικό για γλώσσα IDE, καθώς μπορεί χρησιμοποιηθεί για να γραφτεί κώδικας σε C#, C++, VB (Visual Basic), Python, JavaScript και πολλές άλλες γλώσσες. Παρέχει υποστήριξη για 36 διαφορετικές γλώσσες προγραμματισμού. Είναι διαθέσιμο για Windows καθώς και για macOS. [11]

2.8.2 Γιατί Visual Studio

Παρέχει πολλά εργαλεία που διευκολύνουν τη διαδικασία προγραμματισμού και δοκιμών, για παράδειγμα το εργαλείο εντοπισμού σφαλμάτων (debugger).

Εντοπισμός σφαλμάτων (debuging) σημαίνει να εκτελεί τον κώδικά βήμα προς βήμα χρησιμοποιώντας το εργαλείο Visual Studio, για να βρεί το ακριβές σημείο όπου συμβαίνει το λάθος προγραμματισμού. Με αυτόν τον τρόπο βοηθάει τον προγραμματιστή να καταλάβει ποιες διορθώσεις πρέπει να κάνει στον κωδικα. Επίσης, συχνά τα εργαλεία εντοπισμού σφαλμάτων επιτρέπουν στον προγραμματιστή να κάνει προσωρινές αλλαγές-διορθώσεις στις τιμές των μεταβλητών κατα τη διάρκεια εκτέλεσης του προγράμματος , ώστε να μπορεί να συνεχίσει να εκτελεί, παράγοντας έτσι το επιθυμητό αποτέλεσμα. [12]

2.9 C#

Η ανάπτυξη του software κώδικα για την δημιουργία της εφαρμογής έγινε σε C#. Η εφαρμογή είναι υπεύθυνη για την επεξεργασία των μετρήσεων, την υλοποίηση των απαραίτητων υπολογισμών, την αποστολή τους στη βάση δεδομένων και την εμφάνιση τους σε ένα γραφικό περιβάλλον-UI προς τον χρήστη.

2.9.1 Τί είναι C#

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού. Η C# είναι ανοιχτού κώδικα, απλό, μοντέρνο, ευέλικτο και ευέλικτο

Μια γλώσσα προγραμματισμού στην επιστήμη των υπολογιστών είναι μια γλώσσα που χρησιμοποιείται για τη σύνταξη προγραμμάτων λογισμικού. [13]

2.9.2 Βασικά χαρακτηριστικά αντικειμενοστραφή προγραμματισμού

Τα βασικά χαρακτηριστικά του αντικειμενοστραφή προγραμματισμού είναι τα παρακάτω:

- Κλάσσεις (Classes)
- Κληρονομικότητα (Inheritance)
- Ενθυλάκωση (Encapsulation)
- Πολυμορφισμός (Polymorphism) [13]

2.9.3 Βασικά χαρακτηριστικά C# Γλώσσας Προγραμματισμού

Η C# είναι μια γλώσσα προγραμματισμού που αναπτύχθηκε και ξεκίνησε από τη Microsoft το 2001. Η C# είναι μια απλή, μοντέρνα και αντικειμενοστραφής γλώσσα που παρέχει στους σύγχρονους προγραμματιστές ευελιξία και δυνατότητες για την κατασκευή λογισμικού που όχι μόνο θα λειτουργεί σήμερα αλλά θα ισχύει για χρόνια στο μέλλον.

Τα βασικά χαρακτηριστικά της γλώσσας C # περιλαμβάνουν:

- Μοντέρνα και εύκολη στην εκμάθηση
- Γρήγορη και δωρεάν
- Ευέλικτη και προσαρμόσιμη σε πολλές πλατφόρμες (δίνει τη δυνατότητα)
- Ασφαλής και Αποτελεσματική
- Πολλαπλών Πλατφορμών (Cross platform)
- Εξελίσσεται σύμφωνα με τις ανάγκες του κοινού

2.9.4 C# - Μοντέρνα και εύκολη στην εκμάθηση

Το C# είναι μια απλή, μοντέρνα και αντικειμενοστραφής γλώσσα προγραμματισμού. Ο σκοπός της C# είναι να αναπτύξει μια γλώσσα προγραμματισμού που δεν είναι μόνο εύκολη στην εκμάθηση αλλά και υποστηρίζει τη σύγχρονη λειτουργικότητα για κάθε είδους ανάπτυξη λογισμικού.

Αν κάποιος κοιτάξει την ιστορία των γλωσσών προγραμματισμού και των χαρακτηριστικών τους, κάθε γλώσσα προγραμματισμού σχεδιάστηκε για συγκεκριμένο σκοπό για να λύσει μια συγκεκριμένη ανάγκη εκείνη τη στιγμή.

Ωστόσο, η γλώσσα C # σχεδιάστηκε για να λαμβάνει υπόψη τις ανάγκες των επιχειρήσεων και της αγοράς. Η γλώσσα C # σχεδιάστηκε για τις επιχειρήσεις να δημιουργούν όλα τα είδη λογισμικού χρησιμοποιώντας μία μόνο γλώσσα προγραμματισμού.

Η C# παρέχει λειτουργικότητα για την υποστήριξη της σύγχρονης ανάπτυξης λογισμικού. Η C# υποστηρίζει ανάγκες ανάπτυξης Ιστού, Κινητών και εφαρμογών. Μερικές από τις σύγχρονες δυνατότητες γλώσσας προγραμματισμού που υποστηρίζει η C# είναι γενικά, τύποι var, αυτόματη προετοιμασία τύπων και συλλογών, εκφράσεις λάμδα, δυναμικός προγραμματισμός, ασύγχρονος προγραμματισμός, πλειάδες, αντιστοίχιση μοτίβων, προηγμένος εντοπισμός σφαλμάτων και χειρισμός εξαιρέσεων και πολλά άλλα.

Οι σύνταξη της γλώσσας C# επηρεάζεται από τη C ++, Java, Pascal και μερικές άλλες γλώσσες που είναι εύκολο να υιοθετηθούν. Η C# αποφεύγει επίσης την πολυπλοκότητα και τα μη δομημένα χαρακτηριστικά γλώσσας.[14]

2.9.5 C# - Γρήγορη και ανοικτού κώδικα (open source)

Η C# είναι γρήγορη σε σύγκριση με πολλές άλλες γλώσσες προγραμματισμού υψηλού επιπέδου.

Επίσης, είναι γλώσσα ανοιχτού κώδικα κάτω από το .NET Foundation, το οποίο διέπεται και λειτουργεί ανεξάρτητα από τη Microsoft. Οι προδιαγραφές της γλώσσας C#, οι μεταγλωττιστές και τα σχετικά εργαλεία είναι έργα ανοιχτού κώδικα στο Github. Παρά το γεγονός ότι για την σχεδίαση χαρακτηριστικών γλωσσών C# είναι επικεφαλής της Microsoft, η κοινότητα ανοιχτού κώδικα είναι πολύ δραστήρια στην ανάπτυξη και βελτίωση της γλωσσας. Για αυτο και θεωρήθηκε δόκιμο να αναπτυχθεί η εφαρμογή του προϊόντος σε αυτή τη γλώσσα.

2.9.6 C# - Πολλαπλών Πλατφορμών (Cross platform)

Η C# είναι γλώσσα προγραμματισμού πολλαπλών πλατφορμών. Μπορεί να δημιουργήσει εφαρμογές .NET που μπορούν να αναπτυχθούν σε πλατφόρμες Windows, Linux και Mac. Οι εφαρμογές C# μπορούν επίσης να αναπτυχθούν σε cloud. Κατ’ επέκταση δίνει τη δυνατότητα και στο ίδιο το προϊόν να επεκταθεί σε άλλες πλατφόρμες. [14]

2.9.7 C# Ασφαλής και Αποτελεσματική

Η C# επικεντρώνεται επίσης στη σύνταξη αποτελεσματικού κώδικα. Επιπλόν, η C# είναι μια ασφαλής γλώσσα τύπου. Η C# δεν επιτρέπει μετατροπές τύπου που μπορεί να οδηγήσουν σε απώλεια δεδομένων ή άλλα προβλήματα. Η C# επιτρέπει στους προγραμματιστές να γράφουν ασφαλή κώδικα. Επιπλέον, έχει αναπτύξει πολλά εργαλεία για αυθεντικοποίηση και δίνει τη δυνατότητα στους προγραμματιστές να ασφαλίσουν τις εφαρμογές που αναπτύσσουν. Κάτι που θα ήταν απαραίτητο να έχει προβλεφθεί στην περίπτωση που το προϊόν που αναπτύχθηκε στην παρούσα διπλωματική ήταν επιθυμητό να πωληθεί στην σύγχρονη αγορά. [14]

2.9.8 C# σαν πολυεργαλείο

Η C# είναι σαν πολυεργαλείο. Ενώ οι περισσότερες γλώσσες προγραμματισμού σχεδιάστηκαν για συγκεκριμένο σκοπό, η C# σχεδιάστηκε για να κάνει C#. Μπορεί να χρησιμοποιηθεί η C# για να δημιουργηθούν σύγχρονες εφαρμογές λογισμικού. Η C# μπορεί να χρησιμοποιηθεί για την ανάπτυξη όλων των ειδών εφαρμογών, συμπεριλαμβανομένων εφαρμογών πελάτη Windows, στοιχείων και βιβλιοθηκών, υπηρεσιών και API, εφαρμογών Ιστού, εφαρμογών για κινητά, εφαρμογών cloud και βιντεοπαιχνιδιών. Επομένως, παρέχει τη δυνατότητα επέκτασης του προϊόντος είτε σε online εφαρμογή είτε σε εφαρμογή κινητού[14]

Συμπερασματικά, για τους παραπάνω λόγους και θεωρήθηκε σκόπιμο να αναπτυχθεί η εφαρμογή του προϊόντος σε αυτή τη γλώσσα.

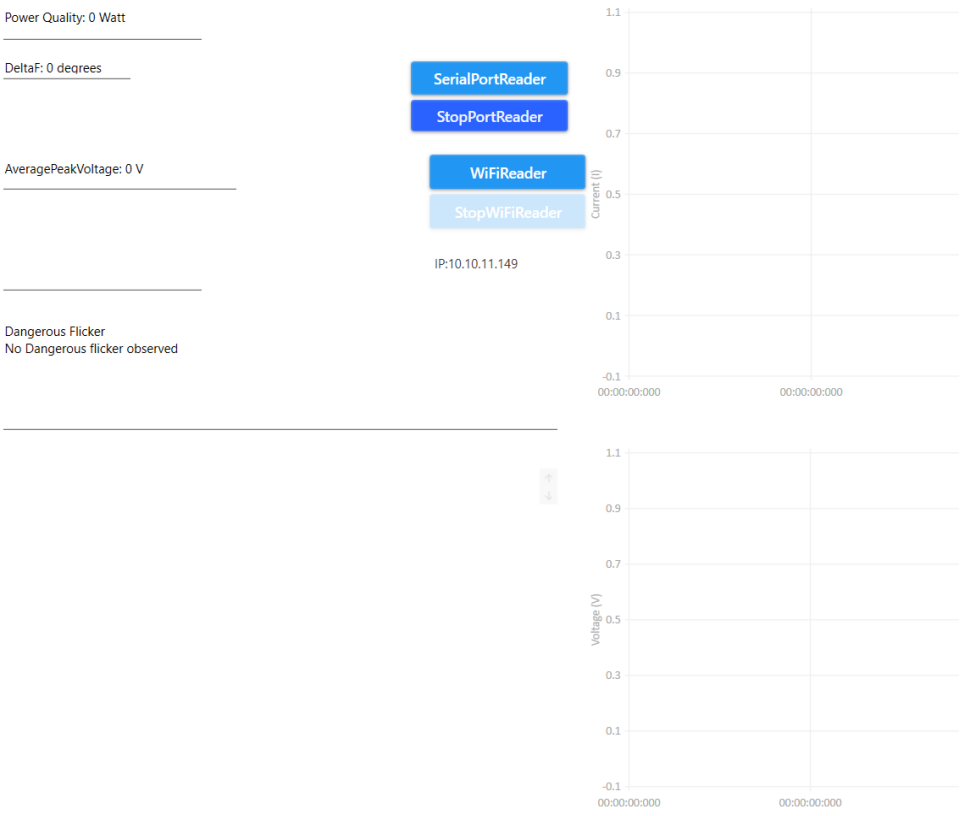
2.10 User Interface (UI)

Για την δημιουργία του User Interface χρησιμοποιήθηκε WPF.

2.10.1Windows Presentation Foundation (WPF)

Το WPF, το οποίο σημαίνει το Windows Presentation Foundation, είναι η τελευταία προσέγγιση της Microsoft σε ένα πλαίσιο GUI, που χρησιμοποιείται με το .NET framework.

Αλλά τι είναι ένα πλαίσιο GUI; Το GUI σημαίνει γραφικό περιβάλλον αναπαράστασης χρήστη και παρακάτω φαίνεται ένα στιγμιότυπο-παράδειγμα από το στο κύριο παράθυρο της διπλωματικής.



Σχήμα 2.6: Παράδειγμα του UI .

Τα Windows διαθέτουν GUI για εργασία στον υπολογιστή . Ένα πλαίσιο GUI επιτρέπει να δημιουργηθεί μια εφαρμογή με ένα ευρύ φάσμα στοιχείων GUI, όπως ετικέτες, πλαίσια κειμένου και άλλα γνωστά στοιχεία. Χωρίς ένα πλαίσιο GUI θα πρέπει κάποιος να σχεδιάσει αυτά τα στοιχεία χειροκίνητα και να χειριστεί όλα τα σενάρια αλληλεπίδρασης χρήστη, όπως για παράδειγμα η εισαγωγή κειμένου, τα κουμπιά και λοιπά. Αυτό είναι πολύ δουλειά, επομένως, οι περισσότεροι προγραμματιστές θα χρησιμοποιήσουν ένα πλαίσιο GUI το οποίο θα κάνει όλη τη βασική εργασία και θα επιτρέπει στους προγραμματιστές να επικεντρωθούν στην κατασκευή εξαιρετικών εφαρμογών.

Υπάρχουν πολλά πλαίσια GUI εκεί έξω, αλλά για προγραμματιστές .NET, τα πιο ενδιαφέροντα είναι επί του παρόντος τα WinForms και WPF. Το WPF είναι νεότερο από WinForms, αλλά η Microsoft διατηρεί και υποστηρίζει WinForms. Υπάρχουν αρκετές διαφορές μεταξύ των δύο πλαισίων, αλλά ο σκοπός τους είναι ο ίδιος: Να διευκολυνθεί η δημιουργία εφαρμογών με ένα ωραίο GUI.

2.10.2 Πλεονεκτήματα WPF

Τα πλεονεκτήματα του WPF:

- Είναι νεότερο (από το WinForms) και επομένως περισσότερο εναρμονισμένο με τα ισχύοντα πρότυπα
- Η Microsoft το χρησιμοποιεί για πολλά εργαλεία, π.χ. Visual Studio
- Είναι πιο ευέλικτο (από το WinForms), ώστε να μπορεί ο χρήστης να κάνει περισσότερα πράγματα χωρίς να χρειάζεται να γράψει ή να αγοράσει νέα στοιχεία ελέγχου
- Όταν πρέπει να χρησιμοποιηθούν στοιχεία ελέγχου τρίτων, οι προγραμματιστές αυτών των στοιχείων ελέγχου πιθανότατα να επικεντρώνονται περισσότερο στο WPF επειδή είναι νεότερο
- Το XAML διευκολύνει τη δημιουργία και την επεξεργασία του GUI και επιτρέπει την κατανομή της εργασίας μεταξύ ενός σχεδιαστή (XAML) και ενός προγραμματιστή (C #, VB.NET κ.λπ.)
- Βάση δεδομένων, η οποία επιτρέπει να έχει έναν πιο καθαρό διαχωρισμό δεδομένων και προγράμματος-κώδικα
- Χρησιμοποιεί επιτάχυνση υλικού για σχεδίαση του GUI, για καλύτερη απόδοση
- Επιτρέπει να γίνουν διεπαφές χρήστη τόσο για εφαρμογές Windows όσο και για εφαρμογές ιστού (Silverlight / XBAP) [15]

Επομένως, όλα αυτά τα πλεονεκτήματα οδήγησαν στην επιλογή του WPF για τον προγραμματισμό του GUI.

2.10.3 Τί είναι XAML

Το XAML, που σημαίνει eXtensible Application Markup Language, είναι η παραλλαγή της XML της Microsoft για την περιγραφή ενός GUI. Σε προηγούμενα πλαίσια GUI, όπως το WinForms, δημιουργούνταν ένα GUI στην ίδια γλώσσα που θα χρησιμοποιούνταν για αλληλεπίδραση με το GUI, π.χ. C# ή VB.NET και συνήθως συντηρούνταν από τον σχεδιαστή (π.χ. Visual Studio), αλλά με το XAML, η Microsoft πηγαίνει με άλλο τρόπο. Όπως και με το HTML, μπορεί να γράφει και να επεξεργάζεται εύκολα το GUI από έναν συντηρητή. Αυτό δίνει τη δυνατότητα στην εφαρμογή να είναι εύκολο να συντηρηθεί κάτι που κρίνεται επιθυμητό για την ανάπτυξη του παρόντος προϊόντος.

Είτε δημιουργείται ένα παράθυρο είτε μια σελίδα, θα αποτελείται από ένα έγγραφο XAML και ένα αρχείο πίσω από τον κώδικα (CodeBehind), το οποίο δημιουργεί μαζί το παράθυρο/σελίδα. Το αρχείο XAML περιγράφει τη διεπαφή με όλα τα στοιχεία της, ενώ το CodeBehind χειρίζεται όλα τα συμβάντα και έχει πρόσβαση για χειρισμό μέσα από τα χειριστήρια της XAML.

2.10.4 Σύνδεση Δεδομένων (data binding) μέσω WPF

Η σύνδεση δεδομένων είναι (data binding) μια γενική τεχνική που συνδέει δύο πηγές δεδομένων/πληροφοριών μαζί και διατηρεί το συγχρονισμό των δεδομένων.

Η σύνδεση δεδομένων στο WPF είναι ο προτιμώμενος τρόπος μεταφοράς δεδομένων από κώδικα στο επίπεδο UI. Σίγουρα, οι ιδιότητες ρυθμίζονται χειροκίνητα ή ένα ListBox μπορεί να συμπληρωθεί προσθέτοντας στοιχεία σε αυτό από έναν βρόχο, αλλά ο καθαρότερος και αποδοτικότερος τρόπος του WPF είναι να προσθέτει μια σύνδεση μεταξύ της πηγής και του στοιχείου διεπαφής χρήστη προορισμού. Αυτή η τεχνική χρησιμοποιήθηκε ευρύτατα στην ανάπτυξη της εφαρμογής.

2.10.5 Σύνδεση στη Βάση Δεδομένων

Για να αποθηκεύεται τοπικά μια Βάση Δεδομένων, απαιτείται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS). Αυτό σημαίνει ότι το DBMS είναι τρέχει σε έναν υπολογιστή και χρησιμοποιεί το χώρο του υπολογιστή για την αποθήκευση του. Επιπλέον, είναι προσβάσιμο μόνο από συσκευές που είναι συνδεδεμένες στο δίκτυο του υπολογιστή (όχι online).

Ένα σύστημα διαχείρισης βάσης δεδομένων (DBMS) είναι ένα σύνολο σχετικών δεδομένων και ένα σύνολο προγραμμάτων που χρησιμοποιούνται για την πρόσβαση σε αυτά τα δεδομένα. Η συλλογή δεδομένων, που συνήθως αναφέρεται ως βάση δεδομένων, παρέχει πληροφορίες για έναν οργανισμό. Ο κύριος στόχος ενός DBMS είναι να παρέχει έναν τρόπο αποθήκευσης και ανάκτησης πληροφοριών από βάσεις δεδομένων που είναι βολικές και αποτελεσματικές.[16]

Τα συστήματα βάσεων δεδομένων έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να χειρίζονται μεγάλο αριθμό πληροφοριών. Η διαχείριση δεδομένων περιλαμβάνει τόσο την ταχύτητα δομών αποθήκευσης πληροφοριών όσο και την παροχή μηχανισμών χειρισμού πληροφοριών. Επιπλέον, τα συστήματα βάσεων δεδομένων πρέπει να διασφαλίζουν την ασφάλεια των αποθηκευμένων πληροφοριών, ανεξάρτητα από προβλήματα συστήματος ή μη εγκεκριμένες προσπάθειες πρόσβασης. Εάν τα δεδομένα μοιράζονται μεταξύ διαφορετικών χρηστών, το σύστημα θα πρέπει να αποφεύγει πιθανώς εσφαλμένα αποτελέσματα. [16]

Στην περίπτωσή της παρούσας εργασίας, το DBMS που επιλέχθηκε να χρησιμοποιηθεί για την τοπική Βάση Δεδομένων (local DataBase) είναι το XAMPP.

2.11 XAMPP

Το XAMPP είναι ένα πακέτο λογισμικού. Ο τίτλος που χρησιμοποιείται για τη συλλογή των εν λόγω προγραμμάτων και εφαρμογών ελεύθερου λογισμικού. Το όνομα είναι ένα αρκτικόλεξο, με κάθε γράμμα να αντιπροσωπεύει ένα από τα πέντε βασικά στοιχεία. Το πακέτο λογισμικού περιέχει τον διακομιστή ιστού Apache, το σχετικό σύστημα διαχείρισης βάσεων δεδομένων MySQL (ή MariaDB) και τις γλώσσες σεναρίου Perl και PHP. Το αρχικό X σημαίνει τα λειτουργικά συστήματα με τα οποία λειτουργεί: Linux, Windows και Mac OS X. [17]

Ακολουθούν σύντομες περιγραφές των εν λόγω προγραμμάτων.

2.11.1 Apache

Apache: διακομιστής ιστού ανοικτού κώδικα. Ο Apache είναι ο πιο διαδεδομένος διακομιστής παγκοσμίως για την παράδοση περιεχομένου ιστού. Η εφαρμογή διακομιστή διατίθεται ως δωρεάν λογισμικό από το Apache Software Foundation.

2.11.2 MySQL

MySQL / MariaDB: στο MySQL, το XAMPP περιέχει ένα από τα πιο δημοφιλή συστήματα σχεσιακής διαχείρισης βάσεων δεδομένων στον κόσμο. Σε συνδυασμό με τον διακομιστή ιστού Apache και τη γλώσσα δέσμης ενεργειών PHP, η MySQL προσφέρει αποθήκευση δεδομένων για υπηρεσίες διαδικτύου. Οι τρέχουσες εκδόσεις XAMPP έχουν αντικαταστήσει το MySQL με το MariaDB (που αναπτύχθηκε από την κοινότητα του έργου MySQL, κατασκευασμένο από τους αρχικούς προγραμματιστές). [17]

2.11.3 PHP

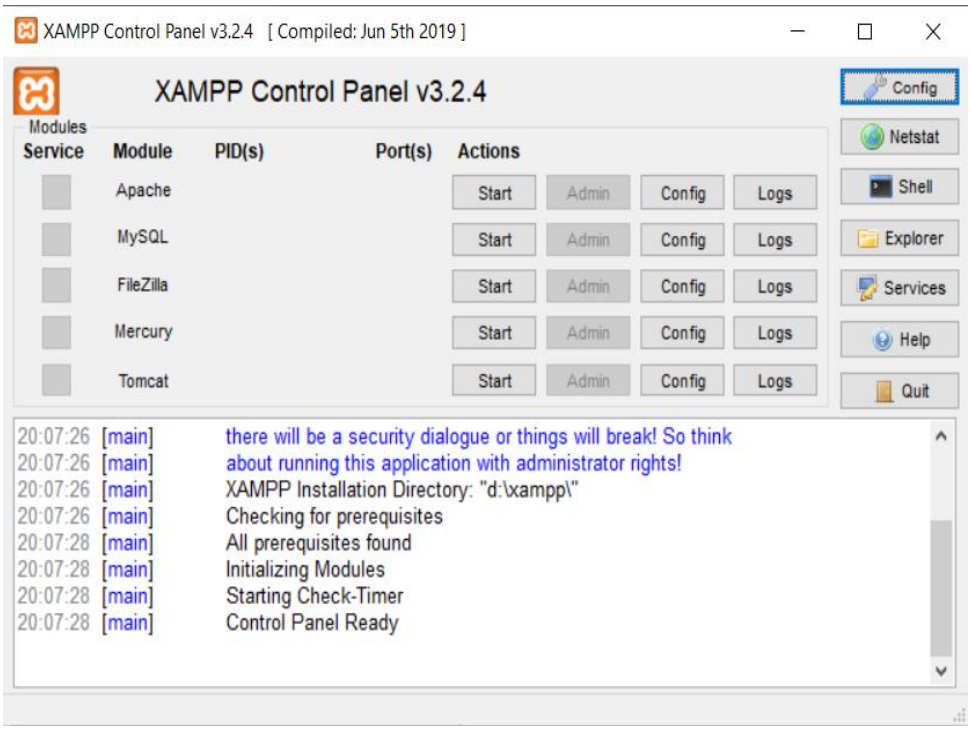
PHP: η γλώσσα προγραμματισμού από την πλευρά του διακομιστή. Η PHP επιτρέπει στους χρήστες να δημιουργούν δυναμικούς ιστότοπους ή εφαρμογές. Η PHP μπορεί να εγκατασταθεί σε όλες τις πλατφόρμες και υποστηρίζει πολλά διαφορετικά συστήματα βάσεων δεδομένων. [17]

2.11.4 Εφαρμογές του XAMPP

Ένας διακομιστής XAMPP μπορεί να εγκατασταθεί και να χρησιμοποιηθεί με ένα και μόνο εκτελέσιμο αρχείο γρήγορα και εύκολα, λειτουργεί ως τοπικό σύστημα δοκιμών για Linux, Windows και Mac OS X. Το πακέτο λογισμικού περιέχει τα ίδια στοιχεία που βρίσκονται σε κοινούς διακομιστές ιστού. Οι προγραμματιστές έχουν την ευκαιρία να δοκιμάσουν τα έργα τους τοπικά και να τα μεταφέρουν εύκολα σε παραγωγικά συστήματα. Ωστόσο, το XAMPP δεν είναι κατάλληλο για χρήση ως δημόσιος διακομιστής, επειδή πολλές λειτουργίες ασφαλείας έχουν παραμείνει σκόπιμα για να απλοποιήσουν και να επιταχύνουν το σύστημα για δοκιμές.

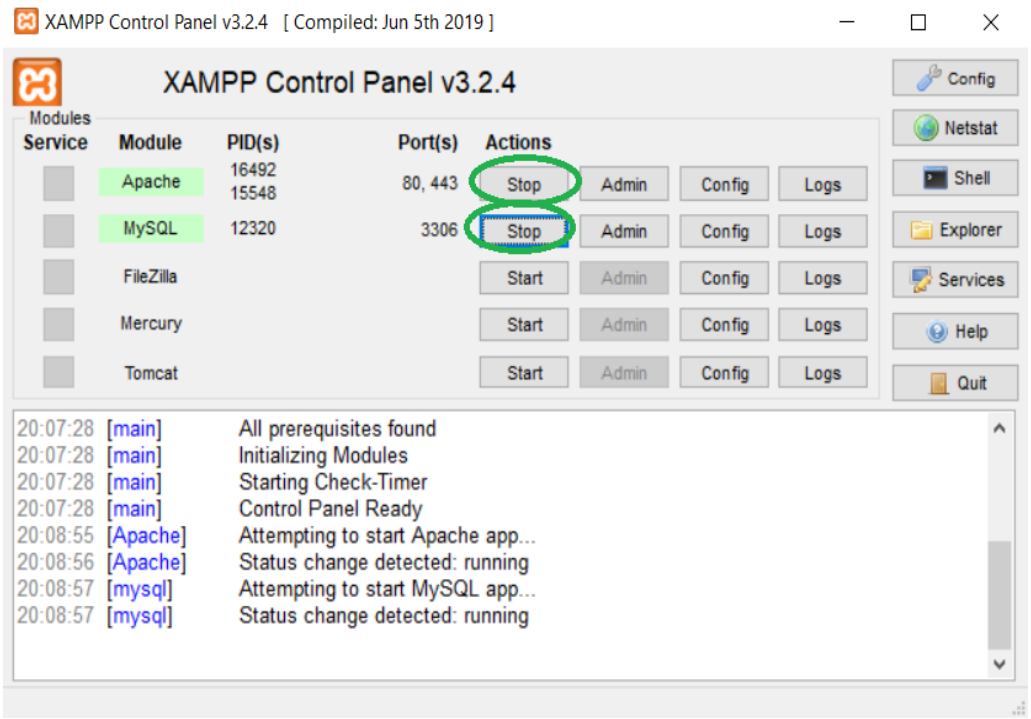
2.11.5 Πρακτική επεξήγηση χρήσης του XAMPP

Η χρήση του XAMPP για τους σκοπούς της παρούσας εργασίας επεξηγείται σε αυτή την παράγραφο. Ανοίγοντας τον XAMPP το UI που εμφανίζεται είναι το εξής:



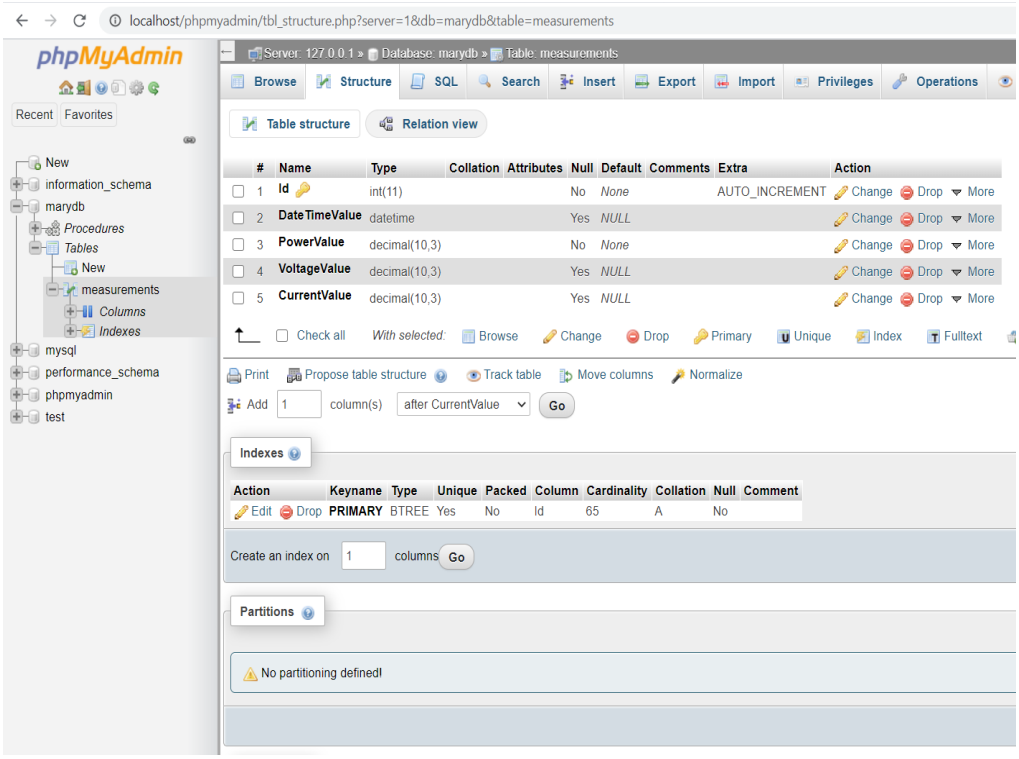
Σχήμα 2.6: UI του XAMPP.

Για να αποκτηθεί πρόσβαση στην τοπική βάση δεδομένων, αυτές οι δύο ιδιότητες πρέπει να οριστούν σε λειτουργία εκκίνησης. Με αυτόν τον τρόπο, το πρόγραμμα της C# μπορεί να πραγματοποιήσει τη σύνδεση (connection) με την local DataBase και να κάνει ότι ενέργειες χρειάζεται όπως εισαγωγή διαγραφή, ανάκτηση δεδομένων (retrieve data) και λοιπά.



Σχήμα 2.7: UI του XAMPP όταν Apache και MySQL είναι ενεργοποιημένα.

Για να αποκτηθεί πρόσβαση στη δημιουργημένη βάση δεδομένων και να ελεγχθούν τα περιεχόμενα της, πρέπει να γίνει κλήση της διεύθυνσης σε ιστότοπο URL (http://localhost/phpmyadmin/tbl_structure.php?server=1&db=marydb&table=measurements). Σημειώνεται ότι η διεύθυνση URL συμπεριλαμβάνοντας τη λέξη-κλειδί "localhost" υποδηλώνει ότι το dataBase αποθηκεύεται τοπικά. Το UI της τοπικής Βάσης Δεδομένων εμφανίζεται παρακάτω.



Σχήμα 2.8: UI της MySQL Database.

2.12 Online DataBase

Το XAMPP είναι ένα πολύ χρήσιμο εργαλείο για την τοπική αποθήκευση μιας βάσης δεδομένων. Ωστόσο, είναι προφανές ότι μια διαδικτυακή Βάση Δεδομένων επιτρέπει την πρόσβαση στα δεδομένα από όλους (που μπορούν να συνδεθούν στο Διαδίκτυο) σε αντίθεση με την τοπική Βάση Δεδομένων που παρουσιάστηκε παραπάνω και παρέχει την πρόσβαση μόνο από όσους έχουν πρόσβαση στο τοπικό δίκτυο. Επομένως, η διαδικτυακή Βάση Δεδομένων παρέχει τα πλεονεκτήματα της φορητότητας και της επεκτασιμότητας. Για παράδειγμα, τα δεδομένα θα μπορούσαν να συλλεχθούν από την υπολογιστική συσκευή, αλλά ο χρήστης που τα επεξεργάζεται μπορεί να τα διαχειρίζεται εξ αποστάσεως (δεν χρειάζεται να έχει φυσική πρόσβαση στην υπολογιστική μηχανή). Για αυτό το λόγο, για την παρούσα εφαρμογή θεωρήθηκε σκόπιμο να αναπτυχθεί η δυνατότητα αποθήκευσης της βάσης δεδομένων online.

2.12.1 Χρήση λογισμικού όπως το MySQL Workbench για τη διαχείριση της διαδικτυακής βάσης δεδομένων

Για την εν λόγω εφαρμογή χρησιμοποιήθηκε ένας δωρεάν διακομιστής φιλοξενίας (hosting server) που μπορεί να βρεθεί στον url (“https://www.awardspace.com/”) με σκοπό τη φιλοξενία της διαδικτυακής βάσης δεδομένων. Για λόγους ασφαλείας, οι βάσεις δεδομένων που δημιουργούνται στον διακομιστής φιλοξενίας μπορούν να χρησιμοποιηθούν μόνο μέσω των βοηθητικών προγραμμάτων phpMyAdmin / phpPgAdmin και μέσω σεναρίων που φιλοξενούνται στον διακομιστή φιλοξενίας. Οι συνδέσεις από άλλες πηγές απορρίπτονται. Ως εκ τούτου, τρίτος διαχειριστής βάσεων δεδομένων όπως το “MySQL Workbench” δεν μπορεί να χρησιμοποιηθεί με τη βάση δεδομένων που είναι διαθέσιμη στους διακομιστές του awardspace. [18] Για αυτό το λόγο χρησιμοποιήθηκε η PHP

Επομένως απαιτείται η χρήση της php για τη μεταφορά δεδομένων από το API (που τα έχει συλλέξει τα δεδομένα) προς τη βάση δεδομένων. Περισσότερες λεπτομέρειες για την υλοποίηση θα συζητηθούν στο κεφάλαιο 3. Ωστόσο, στην επόμενη παράγραφο ακολουθεί μια θεωρητική εξήγηση της PHP.

2.12.2 PHP

Ο PHP Hypertext Preprocessor (PHP) είναι μια γλώσσα προγραμματισμού που επιτρέπει στους προγραμματιστές ιστού να δημιουργούν δυναμικό περιεχόμενο που αλληλεπιδρά με βάσεις δεδομένων. Η PHP χρησιμοποιείται βασικά για την ανάπτυξη εφαρμογών λογισμικού που βασίζονται στον Ιστό.

Μερικά από τα βασικά πλεονεκτήματα που οδήγησαν στη χρήση της PHP είναι:

- Η PHP είναι μια γλώσσα δέσμης ενεργειών διακομιστή που είναι ενσωματωμένη σε HTML. Χρησιμοποιείται για τη διαχείριση δυναμικού περιεχομένου, βάσεων δεδομένων, παρακολούθησης συνεδριών, ακόμη και για τη δημιουργία ολόκληρων ιστότοπων ηλεκτρονικού εμπορίου.

- Είναι ενσωματωμένη σε μια σειρά από δημοφιλείς βάσεις δεδομένων, όπως MySQL, PostgreSQL, Oracle, Sybase, Informix και Microsoft SQL Server.

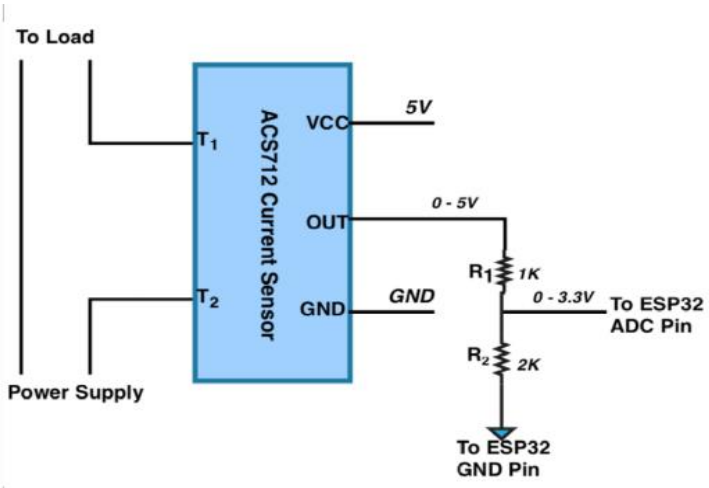
- Η PHP ειναι ευκολη στην εκτέλεση της, ειδικά όταν μεταγλωττίζεται ως μονάδα του Apache.

- Η PHP υποστηρίζει μεγάλο αριθμό σημαντικών πρωτοκόλλων, όπως POP3, IMAP και LDAP. Η PHP πρόσθεσε υποστήριξη για C# και κατανεμημένες αρχιτεκτονικές αντικειμένων

3 Ανάπτυξη του προϊόντος Energy Meter

Για την ανάπτυξη του προϊόντος μετρητή ενέργειας, οι αισθητήρες ρεύματος και τάσης συνδέθηκαν ηλεκτρικά με τον μικροελεγκτή Esp32 και με το πρόσθετο. Το Esp32 ήταν υπεύθυνο για την αποστολή δεδομένων στο API που βρίσκεται σε έναν υπολογιστή. Το API είναι υπεύθυνο για όλους τους υπολογισμούς και την αποστολή δεδομένων στη βάση δεδομένων και στην αναπαράσταση διεπαφής χρήστη. Το τρέχον κεφάλαιο περιέχει την εξήγηση για αυτήν την εφαρμογή.

3.1 Αισθητήρας Ρεύματος



Σχήμα 3.1: Voltage Divider to the output of Current Sensor[19].

Το OUT του τρέχοντος αισθητήρα ACS712 συνδέεται με το ESP32 μέσω ενός διαιρέτη τάσης όπως φαίνεται στο σχήμα 3.1. Ο λόγος είναι ότι το ESP32 διαβάζει δεδομένα στα 3.3V ενώ ο ακροδέκτης OUT (του τρέχοντος αισθητήρα) μεταδίδει (στέλνει) δεδομένα στα 5V.[2] Γι αυτό το λόγο χρησιμοποιήθηκε το κύκλωμα που παρουσιάζεται στην εικόνα 3.1.

3.2 Αισθητήρας Τάσης

Ο αισθητήρας τάσης συνδέεται απευθείας με το ESP32. Το ESP32 διαβάζει δεδομένα στα 3.3V ενώ ο ακροδέκτης OUT (του αισθητήρα τάσης) μεταδίδει (στέλνει) δεδομένα στα 3.3V.

3.2.1 Αποστολή δεδομένων από Esp32 προς το API

Το Esp32 είναι προγραμματισμένο σε C++ σε αρχείο .ino και βαθμονομείται ανάλογα με τις μετρήσεις (βλ. Παράρτημα)

Είναι υπεύθυνη για την ανάγνωση των μετρήσεων των αισθητήρων ρεύματος και τάσης και για την εκτέλεση ορισμένων υπολογισμών. Για παράδειγμα, υπολογίζει τις τιμές RMS του ρεύματος και της τάσης χρησιμοποιώντας τους τύπους (3.1) και (3.2) αντίστοιχα:

$$I_{RMS} = \sqrt{\frac{1}{n}(I^2{}_1 \quad + I^2{}_2 + \ldots + I^2{}_n)} \tag{3.1}$$

$$V_{RMS} = \sqrt{\frac{1}{n}(V^2{}_1 \quad + V^2{}_2 + \ldots + V^2{}_n)} \tag{3.2}$$

όπου I_{RMS} , V_{RMS}είναι οι rms τιμές του ρεύματος και της τάσης αντίστοιχα, I_1, I_2, ... , I_n, V_1, V_2, … , V_nείναι οι μετρήσεις ρεύματος και τάσης αντίστοιχα μιας περιόδου, n είναι ο αριθμός αυτών των μετρήσεων.

Επομένως, η τιμή rms της ισχύος P_{RMS} υπολογίζεται από τον τύπο (3.3):

$$P_{RMS} = I_{RMS} * V_{RMS} \quad (3.3)$$

όπου I_{RMS} , V_{RMS}είναι οι rms τιμές του ρεύματος και της τάσης αντίστοιχα.

Το Esp32 στέλνει στο API μια συμβολοσειρά που περιέχει τις μετρήσεις τάσης Vn, τις μετρήσεις τάσης και την τιμή rms της ισχύος .

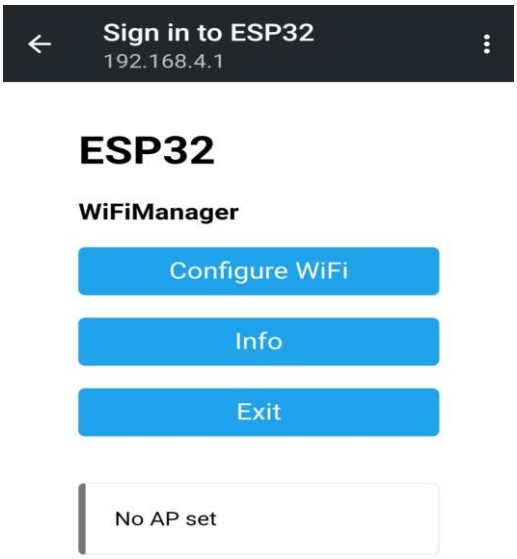
3.3 Ροή επικοινωνίας μεταξύ Esp32 και Api

Η επικοινωνία μεταξύ του Esp32 και του API απαιτεί διαμόρφωση τόσο στην πλευρά Esp32 (μέσω του αρχείου .ino) όσο και στην πλευρά του API. Όπως αναφέρθηκε, η εφαρμογή υποστηρίζει και τις δύο δυνατότητες για την αποστολή δεδομένων είτε μέσω wifi είτε μέσω ενσύρματου καλωδίου USB που συνδέει τον υπολογιστή με ESP. Και οι δύο υλοποιήσεις εξηγούνται σε βάθος στα επόμενα κεφάλαια.

3.3.1 Ρύθμιση Μεταβλητών (Configuration) του Esp32 για Wifi mode

Στο αρχείο .ino ρυθμίζεται η διαμόρφωση του Esp32 για να δημιουργηθεί η επικοινωνία με το Router και το Api.

Αρχικά, χρησιμοποιεί μια βιβλιοθήκη που ονομάζεται <WiFiManager.h>. Χρησιμοποιώντας αυτήν τη βιβλιοθήκη, επιτρέπεται να συνδεθεί αυτόματα με οποιονδήποτε δρομολογητή στην περιοχή ρυθμίζοντας το όνομα χρήστη και τον κωδικό πρόσβασης του δικτύου. Αρχικά ρυθμίζει το Esp32 ως σημείο πρόσβασης. Στη συνέχεια, βρίσκει το δίκτυο esp32 WiFi από οποιαδήποτε συσκευή, επιλέγει το δίκτυο του esp32 WiFi και κάνει βρισκεται σε ένα UI όπως στην εικόνα 3.2. Εκεί επιλέγει το κουμπί “Configure WiFi” και ρυθμίζει το όνομα δικτύου και τον κωδικό πρόσβασης του που θέλει να συνδεθεί το Esp32. Μετά από αυτήν το configuration, το Esp32 συνδέεται με επιτυχία στον επιθυμητό Wifi και στο παγκόσμιο δίκτυο.



Σχήμα 3.2: Configuration του Esp32.

Για να δημιουργηθεί σύνδεση μεταξύ του Api και του Esp32 χρησιμοποιείτε η βιβλιοθήκη <WebServer.h>. Αυτή η βιβλιοθήκη επιτρέπει να μετατραπεί το esp32 ως συσκευή πελάτη που μεταδίδει δεδομένα. Μετά τη διαμόρφωση του δρομολογητή προσπαθεί να συνδεθεί με μία συγκεκριμένη ip και θύρα που λειτουργεί ο διακομιστής. Μετά από μια επιτυχημένη σύνδεση, μπορεί να στείλει δεδομένα στον διακομιστή εκτυπώνοντας δεδομένα στο κανάλι διακομιστή πελάτη που δημιουργήθηκε.

Για να δημιουργηθεί σύνδεση-επικοινωνία μεταξύ του Api και του Esp32 χρησιμοποιεί τη βιβλιοθήκη <WebServer.h>. Αυτή η βιβλιοθήκη επιτρέπει να μετατραπεί το esp32 μας ως client device που μεταδίδει δεδομένα. Μετά τη configuration του router προσπαθεί να συνδεθεί με ένα συγκεκριμένο ip και θύρα που λειτουργεί ο server. Μετά από μια επιτυχημένη σύνδεση, μπορούν να σταλούν δεδομένα στον server εκτυπώνοντας δεδομένα στο (client-server channel) κανάλι διακομιστή-πελάτη που δημιουργήθηκε.

3.3.2 API Configuration για Wifi mode

Η κλάση cServerClass διαχειρίζεται το configuration του Wifi όπως φαίνεται στο σχήμα 3.3 και την αποστολή δεδομένων στο σχήμα 3.4.

```
private void Hearing()
{
    int measurementsCounter = 0;
    DateTime currentDateTime = DateTime.Now;
    while (!IsLiserning == false)
    {

        if (myserver.Pending() == true)
        {
            client = myserver.AcceptTcpClient();
            clientdata = new StreamReader(client.GetStream());
        }

        try
        {
            Message?.Invoke(this, clientdata.ReadLine());
            string data = clientdata?.ReadLine();
            if (data != null)
            {
                System.Console.WriteLine(data);
                if (measurementsCounter < 15)
                {
                    currentDateTime = currentDateTime.AddMilliseconds(100);
                   // measurementsCounter++;
                }
                else
                {
                    currentDateTime = DateTime.Now;
                    measurementsCounter = 0;
                }
                ReadDataFromWiFi(data, measurementsCounter, currentDateTime);
            }
```

Σχήμα 3.3: Κώδικας του API σε C# για διάβασμα δεδομένων μέσω wifi και κλήση της PHPManager.

```
private void ReadDataFromWiFi(string AString,int measurementsCounter, DateTime currentDateTime)
{

    if (IsValidStringReadFromMicrocontroller(AString))
    {
        string[] strList = SplitStringReadFromMicrocontrollerToGetUIPrmsValues(AString);

        Double doubleVoltageValueToParse = Convert.ToDouble(strList[0], CultureInfo.InvariantCulture) + 0.000d;
        Double doubleCurrentValueToParse = Convert.ToDouble(strList[1], CultureInfo.InvariantCulture) + 0.000d;
        Double doublePowerValueToParse = Convert.ToDouble(strList[2], CultureInfo.InvariantCulture) + 0.000d;
        //DateTime currentDateTime = DateTime.Now;

        //UiManagerViewModel.Instance.PowerMeasured = doublePowerValueToParse.ToString();
        UiManagerViewModel.Instance.PowerMeasured = "Power Quality: " + doublePowerValueToParse.ToString() + " Watt";
        ChartCreator.Instance.AddVoltageVarluesInChart(currentDateTime, doubleVoltageValueToParse);
        ChartCreator.Instance.AddCurrentValuesInChart(currentDateTime, doubleCurrentValueToParse);

        //PowerQualityCalculator.Instance.PowerMeasured = Convert.ToSingle(doublePowerValueToParse);
        PowerQualityCalculator.Instance.PowerMeasured = Convert.ToSingle(doublePowerValueToParse);
        ChartCreator.Instance.changePowerMeasured(doubleVoltageValueToParse.ToString());

        MeasurementObject currentMeasurementObject = new MeasurementObject() { DateTimeValue = currentDateTime, PowerVa
        ListOfMeasurmentsObject.Instance.addDataToAllPointsList(currentMeasurementObject);

        ////To Insert online in 90DataBase
        System.Console.WriteLine("Voltage: "+ doubleVoltageValueToParse + " Current: "+ doubleCurrentValueToParse + " P
        //pHPManager.AddValuesToDatabasebyPHPUrl(new MeasurementObject() { DateTimeValue = DateTime.Now, PowerValue = (
        pHPManager.AddValuesToDatabasebyPHPUrl(currentMeasurementObject);
        //PowerQuality.DataContext = ChartCreator.Instance.PowerMeasured;
    }
    Console.WriteLine(AString);
}
```

Σχήμα 3.4: Κώδικας του API σε C# για διάβασμα δεδομένων μέσω wifi και κλήση της PHPManager (συνέχεια).

3.4 Ροή επικοινωνίας μεταξύ Esp32 και Api μέσω Serial Port

Η επικοινωνία μεταξύ του Esp32 και του API απαιτεί ένα configuration τόσο από την πλευρά του Esp32 (μέσω του .ino file) όσο και από την πλευρά του API. όπως έχει αναφερθεί η υλοποίησή υποστηρίζει και τις δύο δυνατότητες, να στέλνονται τα δεδομένα είτε μέσω wifi είτε ενσύρματα μέσω καλωδίου που συνδέεται στον υπολογιστή και με USB στον ESP.

3.4.1 Esp32 Configuration για Serial Port mode

Στο αρχείο .ino ρυθμίζεται το configuration του Esp32 για να δημιουργηθεί η επικοινωνία με το Serial Port και το Api.

3.4.2 Api Configuration για Serial Port mode

Το Api έχει την κλάση SerialPortReader, η οποία είναι υπεύθυνη για τη ρύθμιση της σύνδεσης όσον αφορά το εύρος ζώνης(bandwidth) και το όνομα της θύρας, όπως φαίνεται στο σχήμα 3.5.

```
public SerialPortReader()
{

    pHPManager = new PHPManager();
    listOfMeasurementObject = new List<MeasurementObject>();

    //configuration of arduino, you check if com3 is the port correct,
    //in arduino ide you can make it
    Port = new System.IO.Ports.SerialPort();
    //Port.PortName = "COM4";
    //Port.BaudRate = 115200;

    //for arduino
    Port.PortName = "COM7";
    Port.BaudRate = 9600;

    try
    {
        Port.Open();
    }
    catch (IOException ex){
        MessageBox.Show("IOException:  "+ "  " + ex.Message);
    }
}
```

Σχήμα3.5: Κώδικας του Api σε C#για τη ρύθμιση της σύνδεσης μεταξύ API και Serial Port.

Η μέθοδος SerialPortReader_Load είναι υπεύθυνη για τη δημιουργία ενός νέου thread για να ακούει το API από την σειριακή θύρα συνεχώς μέχρις ότου να κλείσει η θύρα(port). Το κλείσιμο της θύρας γίνεται από το UI πατώντας το κουμπί StopPortReader (η διαδικασία αυτή περιγράφεται σε βάθος στο κεφάλαιο που γίνεται η εξήγηση του UI)

Η μέθοδος SerialPortReader_Load είναι υπεύθυνη για τη δημιουργία νέου νήματος (tread) που ακούει στο serial port μέχρι να κλείσει η πόρτα (port), όπως φαίνεται στο σχήμα 3.6.

```
public void SerialPortReader_Load()
{
    //A Thread to listen forever the serial port
    Thread Hilo = new Thread(ListenSerial);
    Hilo.Start();
}
```

Σχήμα 3.6: Κώδικας του Api σε C# για τη δημιουργία νέου νήματος (tread) που ακούει στο serial port.

Επιπλέον, η κλάση SerialPortReader είναι υπεύθυνη για την επεξεργασία της συμβολοσειράς που έστειλε το ESP32 (με όλες τις χρήσιμες πληροφορίες, που σημαίνει την ακριβή μέτρηση τάσης και ρεύματος αλλά και την υπολογισμένη τιμή RMS) στη μέθοδο ListenSerial, όπως φαίνεται στο σχήμα 3.7.

Επιπρόσθετα, ορίζει το DateTime που έγινε η μέτρηση.

```
private void ListenSerial()
{
    //List<int> list = new List<int>();
    int measurementsCounter = 0;
    DateTime currentDateTime = DateTime.Now;
    while (!IsClosed)
    {

        try
        {
            //read data from arduino
            string AString = Port.ReadLine();

            if (IsValidStringReadFromMicrocontroller(AString))
            {
                string[] strList = SplitStringReadFromMicrocontrollerToGetUIPrmsValues(AString);

                Double doubleVoltageValueToParse = Convert.ToDouble(strList[0], CultureInfo.InvariantCulture) + 0.000d;
                Double doubleCurrentValueToParse = Convert.ToDouble(strList[1], CultureInfo.InvariantCulture) + 0.000d;
                Double doublePowerValueToParse = Convert.ToDouble(strList[2], CultureInfo.InvariantCulture) + 0.000d;
                if (measurementsCounter < 15)
                {
                    currentDateTime=currentDateTime.AddMilliseconds(100);
                   // measurementsCounter++;
                }
                else
                {
                    currentDateTime = DateTime.Now;
                    measurementsCounter = 0;
                }
```

Σχήμα 3.7: Κώδικας του Api σε C# για διάβασμα τιμών από τον Esp32 μέσω SerialPort.

```
UiManagerViewModel.Instance.PowerMeasured = "Power Quality: " + doublePowerValueToParse.ToString() + " Watt";
MeasurementObject currentMeasurementObject = new MeasurementObject() { DateTimeValue = currentDateTime, PowerValue
ListOfMeasurmentsObject.Instance.addDataToAllPointsList(currentMeasurementObject);
ChartCreator.Instance.AddVoltageVarluesInChart(currentDateTime,doubleVoltageValueToParse);
ChartCreator.Instance.AddCurrentValuesInChart(currentDateTime, doubleCurrentValueToParse);

PowerQualityCalculator.Instance.PowerMeasured = Convert.ToSingle(doublePowerValueToParse);
ChartCreator.Instance.changePowerMeasured(doubleVoltageValueToParse.ToString());
////To Insert online in DataBase
pHPManager.AddValuesToDatabasebyPHPUrl(new MeasurementObject() { DateTimeValue = DateTime.Now, PowerValue = (decim




////Insert in DataBase localy
// DataBaseManager.Instance.DBOpenConnection();
//DataBaseManager.Instance.InsertValuesInDataBase(DateTime.Now, decimaPowerValueToParse, decimaVoltageValueToParse
// DataBaseManager.Instance.DBCloseConnection();
```

Σχήμα 3.8: Κώδικας του Api σε C# για διάβασμα τιμών από τον Esp32 μέσω SerialPort. .

Μια άλλη λειτουργικότητα είναι η αποστολή των δεδομένων είτε στην online είτε στην τοπική βάση δεδομένων. Τέλος, ενημερώνει την υπολογισμένη ισχύ RMS και ενώ ταυτόχρονα ανανεώνει δυναμικά το User Interface, όπως φαίνεται στο σχήμα 3.8.

Όπως έχει ειπωθεί ο Esp32 στέλνει τα δεδομένα στο API μέσω μιας συμβολοσειράς (string) όπως φαίνεται σε κομματι του στο .ino αρχείο (ολόκληρο το .ino αρχείο βρίσκεται στο παράρτημα), όπως φαίνεται στο σχήμα 3.9.

```
float U = (voltageSensor.getVoltageAC()-1) *10.5; //calibration

float I = (currentSensor.getCurrentAC()*1000 - 45.76) *3.28; //in mA with calibration

float P = U * I / 1000;

Serial.println(String("U=") + voltage  +"I="+ current +"P=" + P );
```

Σχήμα 3.9: Εντολές από .ino αρχείο που στέλνουν πληροφορία από το Esp32 στο API.

Αντίστοιχα το API έχει μία κλάση που δέχεται-διαβάζει την συμβολοσειρά (με τις μετρήσεις) που έχει αποσταλεί από το Esp32 και την αποκωδικοποιεί. Οι βοηθητικές μέθοδοι για την αποκωδικοποίηση της συμβολοσειράς από το Esp32 είναι το IsValidStringReadFromMicrocontroller που επικυρώνει τη σωστή μορφή της συμβολοσειράς και το SplitStringReadFromMicrocontrollerToGetUIPrmsValues που χωρίζει τα διάφορα συστατικά του string σε substrings, όπως φαίνεται στο σχήμα 3.10.

```
public String[] SplitStringReadFromMicrocontrollerToGetUIPrmsValues(string str) {

        String[] spearator = { "U=", "I=", "P=", "\r" };
        Int32 count = 4;

        String[] strList = str.Split(spearator, count, StringSplitOptions.RemoveEmptyEntries);

        //TODO remove this
        foreach (String s in strList)
        {
            Console.WriteLine(s);
        }


        return strList;
}


private bool IsValidStringReadFromMicrocontroller(string str)
{
    if (str.StartsWith("U=")) return true;
    return false;
}
```

Σχήμα 3.10: Κώδικας του Api σε C# για διάβασμα δεδομένων απο Esp32 .

Τέλος, η κλάση SerialPortReader είναι υπεύθυνη για το κλείσιμο της θύρας (port) και κατά συνέπεια τη σύνδεση με το ESP32 στη μέθοδο SerialPortReader_PortClosed, όπως φαίνεται στο σχήμα 3.11.

```
public void SerialPortReader_PortClosed()
{
    //when the form will be closed this line close the serial port
    IsClosed = true;
    if (Port.IsOpen)
        Port.Close();
}
```

Σχήμα 3.11: Κώδικας του Api σε C# for closing the port.

3.5 PHP manager

Η επικοινωνία μεταξύ του API και της online βάσης δεδομένων για τη μεταφορά των πληροφοριών γίνεται μέσω της κλάσης PHPManager. Συγκεκριμένα, δημιουργήθηκε μια συμβολοσειρά που περιέχει τη διεύθυνση URL που φιλοξενεί την ηλεκτρονική βάση δεδομένων και συνδυάζει τα στοιχεία του αντικειμένου MeasurementObject, συγκεκριμένα τις τιμές PowerValue, VoltageValue, CurrentValue, DateTimeValue.

Η κλάση PHPManager καλείται είτε από το SerialPortREader είτε από το WifiReader, όπως φαίνεται στο σχήμα 3.12.

```
public class PHPManager
{
    string baseUrl= "http://shopmanager.atwebpages.com/accessDB.php";
    //string baseUrl = "http://shopmanager.atwebpages.com/TestJson.php";

    1 reference
    public void AddValuesToDatabasebyPHPUrl(MeasurementObject measurementObject)
    {
        //working url "http://shopmanager.atwebpages.com/accessDB.php?datetime='2020-05-18%2
        string url = baseUrl + "?datetime='"+ measurementObject.DateTimeValue.
            ToString("yyyy-MM-dd hh:mm:ss:fffff") + "'&power="
            + measurementObject.PowerValue.ToString().Replace(",",".") + "&voltage=" +
            measurementObject.VoltageValue.ToString().Replace(",",".") + "&current=" +
            measurementObject.CurrentValue.ToString().Replace(",",".");
       // System.Console.WriteLine(url);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();

        StreamReader streamread = new System.IO.StreamReader(response.GetResponseStream());
        string result = streamread.ReadToEnd();
        streamread.Close();
        System.Console.WriteLine(result);
    }
```

Σχήμα 3.12.: Κώδικας του Api σε C# for sending data through PHP to online DataBase.

Στη συνέχεια, το αρχείο PHP που μεταφορτώνεται στη βάση δεδομένων λαμβάνει τις πληροφορίες από τη διεύθυνση url και τις εισάγει στα αντίστοιχα πεδία του πίνακα measurements της βάσης δεδομένων, όπως φαίνεται στο σχήμα 3.13.

```php
<?php
$servername = "fdb24.awardspace.net";
$username = "3412629_energymeter";
$password = "maryleft9592";
$dbname = "3412629_energymeter";
$datetime = $_GET["datetime"];
$power = $_GET["power"];
$voltage = $_GET["voltage"];
$current =$_GET["current"];
#echo $distance;
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO measurements (DateTimeValue, PowerValue, VoltageValue, CurrentValue)
VALUES ($datetime, $power, $voltage,$current)";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

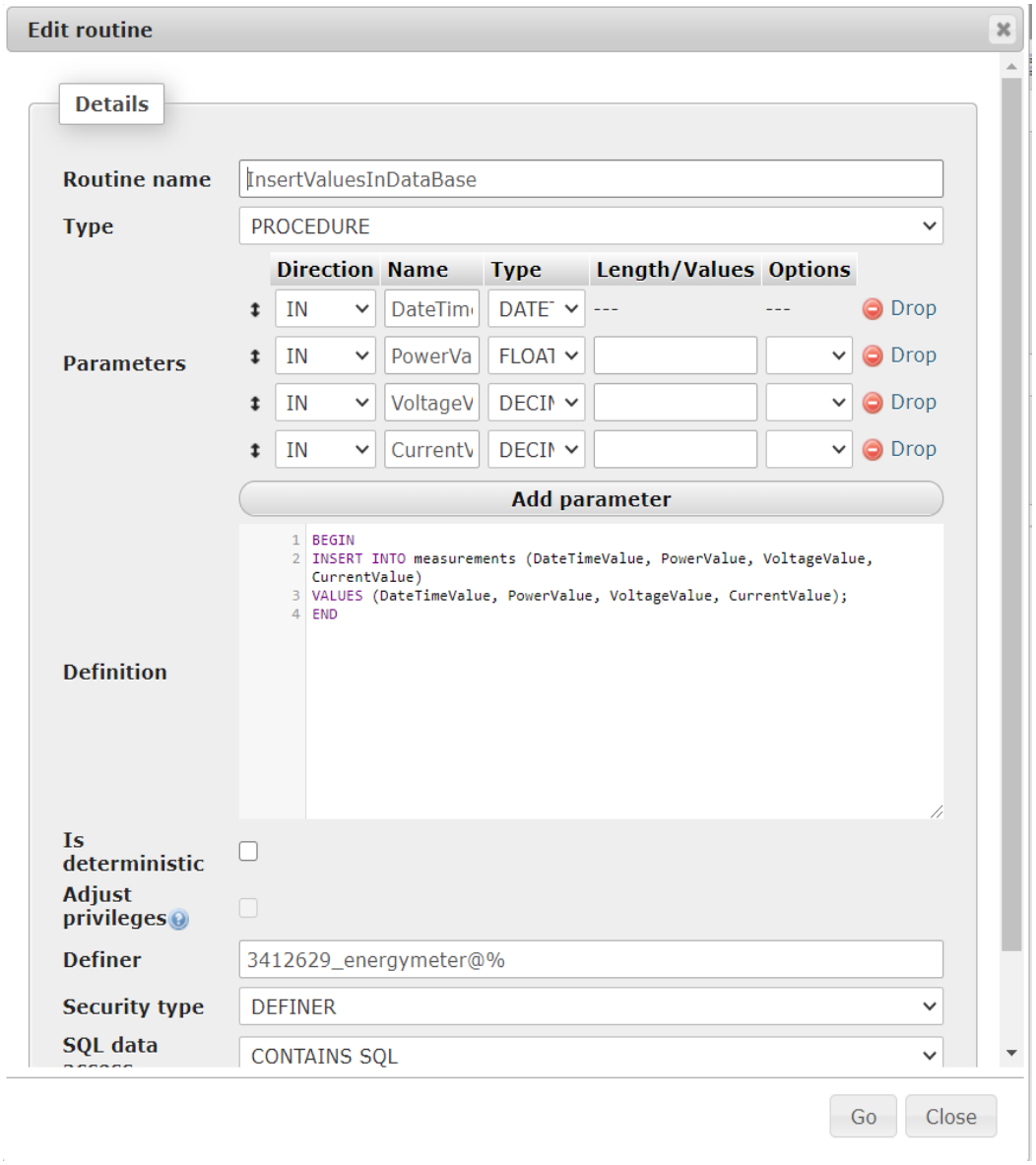
Σχήμα 3.13: PHP code for inserting data through into online DataBase.

3.6 Online DataBase

Η κλάση PHPManager είναι υπεύθυνη για την αποστολή δεδομένων στη διαδικτυακή βάση δεδομένων από το API.

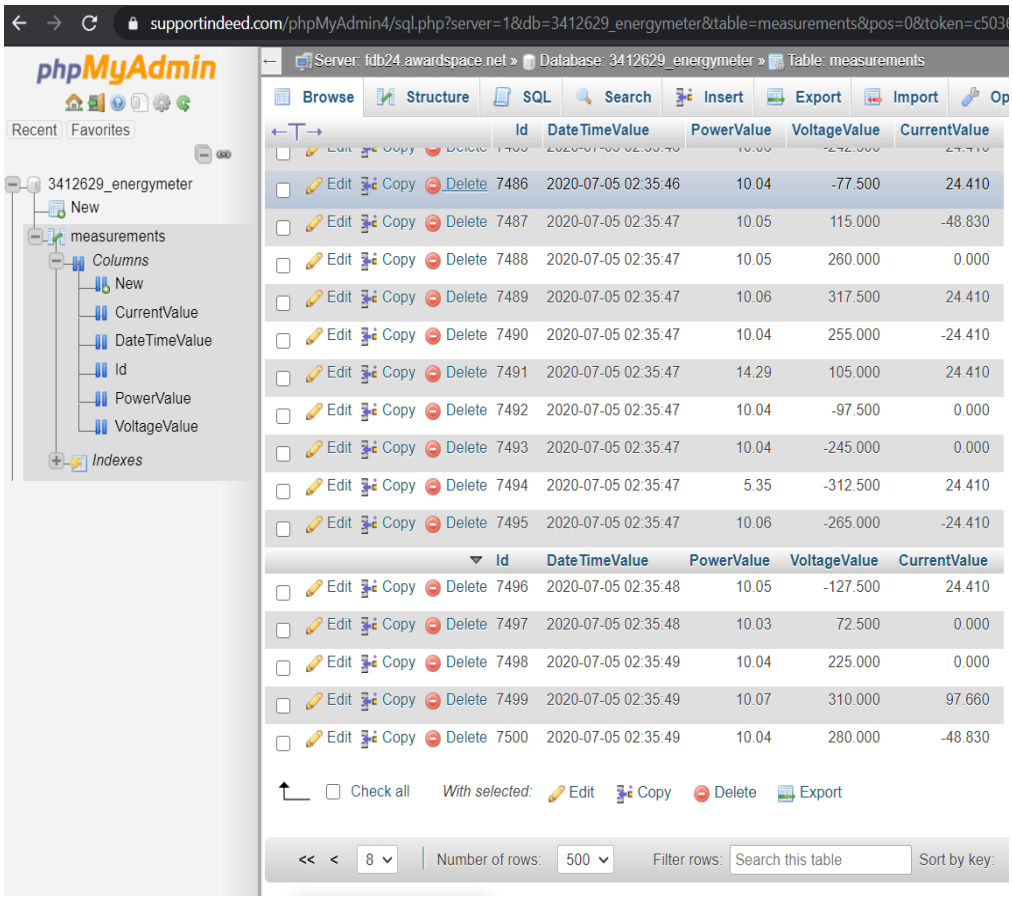
3.6.1 DataBase Query

Το Query που διαχειρίζεται τις πληροφορίες που λαμβάνονται από την PHP και τις εισάγει ως εγγραφή στον πίνακα μετρήσεων βρίσκεται στην ακόλουθη διαδικασία, όπως φαίνεται στο σχήμα 3.14.



Σχήμα 3.14: DataBase Procedure with Query for inserting records.

Ένα παράδειγμα του UI της online Βάσης Δεδομένων με εγγεγραμμένες μετρήσεις παρουσιάζεται στο σχήμα 3.15



Σχήμα 3.15: Online DataBase Example of inserted records.

3.7 Local DataBase

Η κλάση DataBaseManager είναι υπεύθυνη για την αποστολή δεδομένων στη βάση δεδομένων που αποθηκεύονται τοπικά στο XAMPP από το API.

3.7.1 DataBaseManager

Οι ακόλουθες μέθοδοι χρησιμοποιούνται για το άνοιγμα και το κλείσιμο της σύνδεσης της τοπικής βάσης δεδομένων, όπως φαίνεται στο σχήμα 3.16.

```
1 reference
public void DBOpenConnection()
{

    //ΔΗΜΙΟΥΡΓΙΑ ΣΥΝΔΕΣΗΣ
    connectionString = @"server=localhost;userid=root;password=;database=marydb;";

    //cs = "server=db56.grserver.gr;port=3306;userid=wordpress_83;password=tlR593$v;

    connection = new MySqlConnection(connectionString);
    try
    {
        connection.Open();
        //MessageBox.Show("Η συνδεση στην βαση εγινε!");
    }
    catch (MySqlException ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
1 reference
public void DBCloseConnection()
{
    //ΚΛΕΙΣΙΜΟ ΣΥΝΔΕΣΗΣ
    connection.Close();
    //MessageBox.Show("Η συνδεση στην βαση ΕΚΛΕΙΣΕ!");

}
```

Σχήμα 3.16: DataBase Procedure with Query for inserting records.

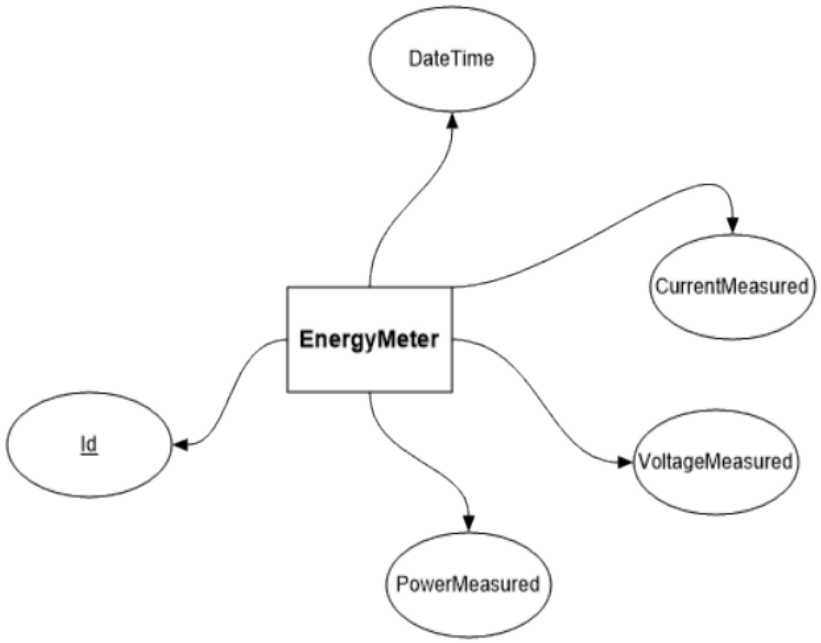
3.7.2 DataBase Structure

Η βάση δεδομένων ονομάζεται mary_dB και ο πίνακας που περιέχει τις μετρήσεις ονομάζεται “measurements”. Έγινε ο σχεδιασμός της βάσης δεδομένων σε μοντέλο ER καθώς είναι ακαδημαϊκά ορθός αλλά και πολύ αποτελεσματικός τρόπος αναπαράστασης.

Ένα μοντέλο οντότητας-σχέσης ή Entity–relationship (μοντέλο ER) περιγράφει τη δομή μιας βάσης δεδομένων με τη βοήθεια ενός διαγράμματος, το οποίο είναι γνωστό ως Entity Relationship Diagram (ER Diagram ή Διάγραμμα ER). Ένα μοντέλο ER είναι ένα σχέδιο ή σχεδιάγραμμα μιας βάσης δεδομένων που μπορεί αργότερα να εφαρμοστεί ως βάση δεδομένων. Τα κύρια συστατικά του μοντέλου E-R είναι: σύνολο οντοτήτων και σύνολο σχέσεων (entity set and relationship set).

Ένα διάγραμμα ER δείχνει τη σχέση μεταξύ συνόλων οντοτήτων. Ένα σύνολο οντοτήτων είναι μια ομάδα παρόμοιων οντοτήτων και αυτές οι οντότητες μπορούν να έχουν χαρακτηριστικά όπως πρωτεύον κλειδί, απαραίτητο πεδίο και λοιπά . Όσον αφορά το DBMS, μια οντότητα είναι ένας πίνακας ή ένα χαρακτηριστικό ενός πίνακα στη βάση δεδομένων, οπότε δείχνοντας τη σχέση μεταξύ πινάκων και των χαρακτηριστικών τους, το διάγραμμα ER δείχνει την πλήρη λογική δομή μιας βάσης δεδομένων.[16]

Ο πίνακας της Βάσης Δεδομένων “measurements” αποτελείται από πέντε στήλες, όπως φαίνεται στο ακόλουθο διάγραμμα ER, στο σχήμα 3.17:



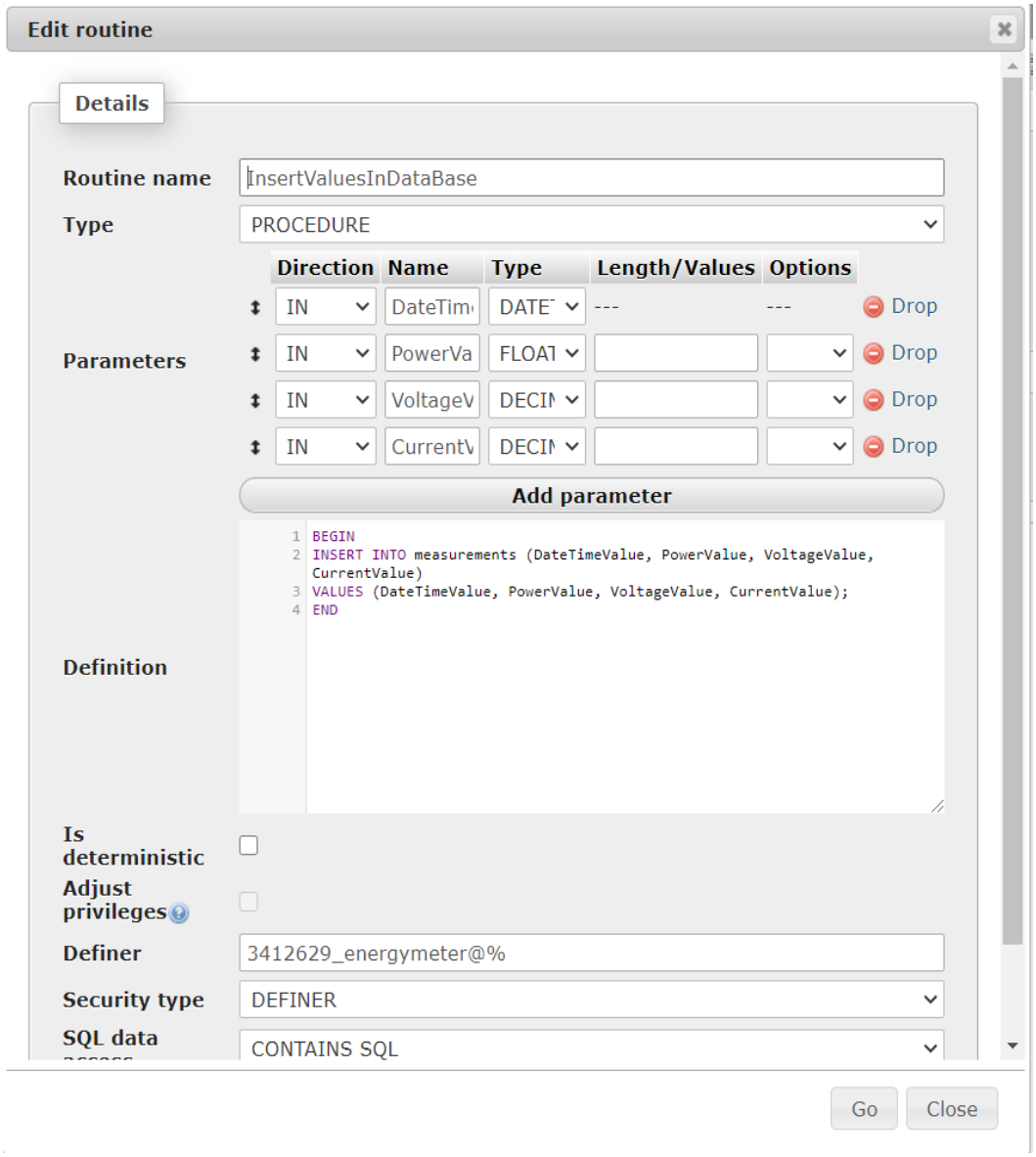
Σχήμα 3.17: ER diagram of the table.

Οι εγγραφές αποτελούνται από τα πεδία:

- Id: primary key εγγραφής
- DateTime : την ακριβή ημερομηνία και ώρα που καταγράφηκε η μέτρηση
- VoltageMeasured: AC τάση που μετρήθηκε
- CurrentMeasured: AC ρεύμα που μετρήθηκε
- PowerMeasured : η RMS τιμή της ισχύς υπολογίζεται μεσω του esp32 (ως πολλαπλασιασμός τιμών RMS ρεύματος και τάσης.)

3.7.3 DataBase Query

Το Query το οποίο διαχειρίζεται την πληροφορία που πήρε από το API και την εισάγει ώς εγγραφή στον πίνακα measurements είναι στο παρακάτω procedure στο σχήμα 3.18



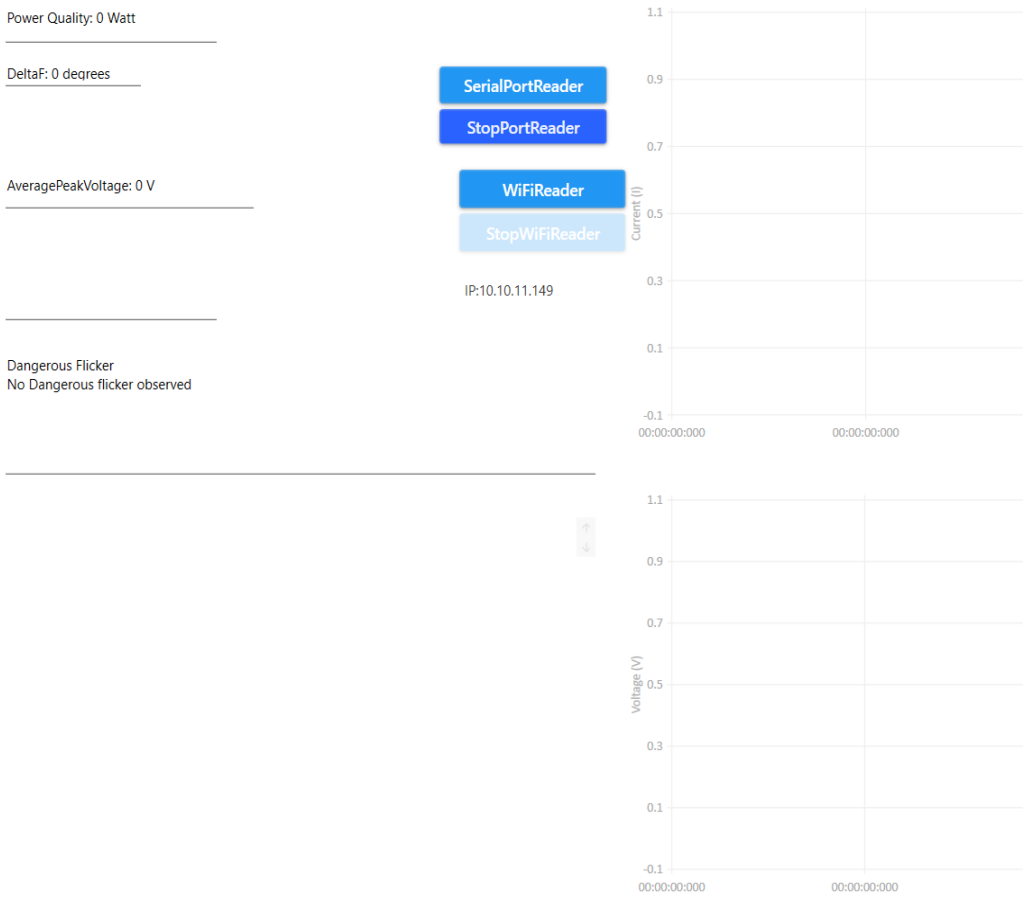
Σχήμα 3.18: DataBase Procedure with Query for inserting records.

3.8 User Interface

Η αναπαράσταση των αποτελεσμάτων σε ένα κατανοητό και φιλικό προς το χρήστη περιβάλλον αποτελεί βασικό μέρος της υλοποίησης. Αναπτύχθηκε σύμφωνα με τα πρότυπα και τις αρχές σχεδιασμού του WPF.

Όπως αναφέρθηκε προηγουμένως, το XAML, που σημαίνει eXtensible Application Markup Language, είναι η παραλλαγή της XML της Microsoft για την περιγραφή ενός GUI.

Στην παρούσα εργασία παρουσιάζεται παρακάτω ένα παράδειγμα του User Interface:



Σχήμα 3.19: Παράδειγμα του UI .

Η περιγραφή των χαρακτηριστικών UI είναι:

- Power Quality : label που αλλάζει δυναμικά ανάλογα με τους υπολογισμούς API για την τιμή RMS ισχύος.
- DeltaPhi :label που αλλάζει δυναμικά ανάλογα με τους υπολογισμούς API και υπολογίζει τη διαφορά φάσης των κυματομορφων τάσης και ρεύματος
- AveragePeakVoltage : label που αλλάζει δυναμικά ανάλογα με τους υπολογισμούς API,
- Dangerous Flicker : που αλλάζει δυναμικά από το API ανάλογα με το αν έχει εντοπιστεί ένα επικίνδυνο flicker
- Overshoot : που αλλάζει δυναμικά από το API ανάλογα με το αν εντοπίστηκε overshoot
- Warnings: που αλλάζει δυναμικά από το API ανάλογα με το αν έχει εντοπιστεί ένδειξη μη καλής ποιότητας ενέργειας
- SerialPortReader : Ένα κουμπί για να ξεκινήσει η σειριακή ανάγνωση των μετρήσεων
- StopPortReader: Ένα κουμπί για να σταματήσει η σειριακή ανάγνωση των μετρήσεων
- WiFiReader: Ένα κουμπί για να ξεκινήσει η ανάγνωση των μετρήσεων μέσω Wifi
- StopWiFiReader: Ένα κουμπί για να σταματήσει η ανάγνωση των μετρήσεων μέσω Wifi
- IP : Μια ετικέτα που αναγνωρίζει την IP στη οποία έχει συνδεθεί το esp32
- Μια δυναμική γραφική αναπαράσταση των μετρήσεων τάσης με την πάροδο του χρόνου, V(t)

- Μια δυναμική γραφική αναπαράσταση των μετρήσεων ρεύματος με την πάροδο του χρόνου, I(t)

Ο κώδικας για τα textboxes, labels και buttons που χρησιμοποιήθηκαν περιγράφονται στα παρακάτω σχήματα 3.20 και 3.21:

```
<TextBox Name="PowerQuality" HorizontalAlignment="Left" Height="33" Margin="21,13,0,0"
        TextWrapping="Wrap" Text="{Binding PowerMeasured,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
        VerticalAlignment="Top" Width="187" />
<TextBox Name="AveragePeakVoltage" HorizontalAlignment="Left" Height="32" Margin="21,154,0,0"
        TextWrapping="Wrap" Text="{Binding AveragePeakVoltage,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
        VerticalAlignment="Top" Width="220" RenderTransformOrigin="0.518,-0.165" />

<TextBox Name="Warnings" HorizontalAlignment="Left" Height="74" Margin="21,207,0,0"
        TextWrapping="Wrap" Text="{Binding Warnings,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
        VerticalAlignment="Top" Width="187"/>
<TextBox Name="DangerousFlicker" HorizontalAlignment="Left" Height="105" Margin="21,306,0,0"
        TextWrapping="Wrap" Text="Dangerous Flicker (with ObservedTime)" VerticalAlignment="Top" Width="523"

<TextBox Name="OvershootDetected" HorizontalAlignment="Left" Height="355" Margin="21,443,0,0"
        TextWrapping="Wrap" Text="{Binding OvershootDetected,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}"
        VerticalScrollBarVisibility="Visible" VerticalAlignment="Top"  Width="523"/>

<Button Name="WiFiConnectBtn" Content="WiFiReader" HorizontalAlignment="Left" Margin="424,154,0,0"
       VerticalAlignment="Top" Width="147" Click="WiFiReader"/>
<Button Name="WiFiDisconnectBtn" Content="StopWiFiReader" HorizontalAlignment="Left" Margin="424,191,0,0"
       VerticalAlignment="Top" Width="147" Click="StopWiFiReader"/>
<Label Name="IPLabel" Content="Label" HorizontalAlignment="Left" Margin="424,243,0,0"
      VerticalAlignment="Top" Width="147"/>
<TextBox Name="DF" HorizontalAlignment="Left" Height="23" Margin="21,60,0,0" TextWrapping="Wrap"
        VerticalAlignment="Top" Width="120"
        Text="{Binding PowerCoefficient,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}" />
```

Σχήμα 3.20: XML code in Xaml for textboxes, labels, buttons.

Προκειμένου να εφαρμοστεί η δυναμική γραφική αναπαράσταση των μετρήσεων τάσης και ρευμάτων με την πάροδο του χρόνου, χρησιμοποιήθηκε ένα πακέτο nuget με το όνομα LiveCharts. Ο κωδικας φαίνεται παρακάτω:

```
<Grid Name="CurrentChart" Margin="213.2,0,-0.4,114" Grid.Column="2" Grid.RowSpan="2">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="484*"/>
        <ColumnDefinition Width="469*"/>
    </Grid.ColumnDefinitions>
    <lvc:CartesianChart Series="{Binding SeriesCollectionCurrent}"
                        LegendLocation="Right" Margin="0,10,-1.2,0" Grid.ColumnSpan="2" >
        <lvc:CartesianChart.AxisX>
            <lvc:Axis Title="Time (t)" LabelFormatter="{Binding YFormatter}"></lvc:Axis>
        </lvc:CartesianChart.AxisX>
        <lvc:CartesianChart.AxisY>
            <lvc:Axis Title="Current (I)" Labels="{Binding Labels}"></lvc:Axis>
        </lvc:CartesianChart.AxisY>
    </lvc:CartesianChart>
</Grid>
<Grid Name="distanceChart" Margin="213.2,412,-0.4,0.4" Grid.Column="2" Grid.RowSpan="4">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="484*"/>
        <ColumnDefinition Width="469*"/>
    </Grid.ColumnDefinitions>
    <lvc:CartesianChart Series="{Binding SeriesCollection}"
                        LegendLocation="Right" Margin="0,10,-1.2,0" Grid.ColumnSpan="2" >
        <lvc:CartesianChart.AxisX>
            <lvc:Axis Title="Time (t)" LabelFormatter="{Binding YFormatter}"></lvc:Axis>
        </lvc:CartesianChart.AxisX>
        <lvc:CartesianChart.AxisY>
            <lvc:Axis Title="Voltage (V)" Labels="{Binding Labels}"></lvc:Axis>
        </lvc:CartesianChart.AxisY>
    </lvc:CartesianChart>
```

Σχήμα 3.21: XML code in Xaml for graphs .

3.9 Υπολογισμοί στο API

Το API είναι υπεύθυνο για πολλούς υπολογισμούς που παρουσιάζονται στο UI.

3.9.1 Υπέρταση ή Πτώση Τάσης και υπολογισμοί

Για τον υπολογισμό της υπέρτασης ή της πτώσης της τάσης υπάρχουν ορισμένες υπολογιστικές μέθοδοι που εφαρμόστηκαν. Πρώτα απ 'όλα για κάθε κορυφή της τάσης που μετριέται προστίθεται σε ένα συνολικό άθροισμα των κορυφών όπως επίσης και ένας μετρητής όλων των κορυφών που προστέθηκαν. Χρησιμοποιώντας τον τύπο:

$$\text{MO} = (V_1 + V_2 + \ldots + V_n) / \text{n} \quad (3.4)$$

όπου MO είναι ο Μέσος Όρος των τοπικών μεγίστων (peak) της τάσης V_1, V_2, … , V_nείναι τα τοπικά μέγιστα της τάσης αντίστοιχα, n είναι ο αριθμός αυτών των μεγίστων που συμπεριλαμβάνονται στην εξίσωση σχηματίζεται ένας μέσος όρος των μέγιστων τάσεων όπως παρουσιάζεται στην εικόνα 3.22. Κάθε εισερχόμενη μέτρηση ελέγχεται εάν βρίσκεται εντός του εύρους 10% του μέσου όρου. Εάν όχι, θεωρείται ως υπέρτασης ή πτώσης της τάσης. αυτός ο έλεγχος πραγματοποιείται στην μέθοδο που εμφανίζεται στην εικόνα 3.23. Εάν εντοπιστεί υπέρτασης, εμφανίζεται ως μήνυμα στο User Interface. Αυτή η μέθοδος έχει υλοποιηθεί όπως φαίνεται στην εικόνα 3.24

```
public double SumNegativePeaksOfVoltage = 0;
public double AverageNegativePeakOfVoltage = 0;
0 references
public void AddMeasurementToPeakAverage(MeasurementObject measurementObject)
{
    numberOfMeasures++;
    SumPeaksOfVoltage += (double)measurementObject.VoltageValue;
    AveragePeakOfVoltage = SumPeaksOfVoltage / numberOfMeasures;

}
```

Σχήμα 3.22: Μέθοδος που προσθέτει την νέα μέτρηση τάσης στον Μέσο Όρο.

```
1 reference
public bool FoundOvershootOrDropOfVoltage(MeasurementObject measurementObject){
    if (AveragePeakOfVoltage * 1.1 < Math.Abs((double)measurementObject.VoltageValue)
        || Math.Abs((double)measurementObject.VoltageValue) < AveragePeakOfVoltage * 0.9) {
        return true;
    }
    return false;
}
```

Σχήμα 3.23: Μέθοδος για εντοπισμό Overshoot ή Dropdown τάσης.

```
public void AddMeasurementToAverage(MeasurementObject measurementObject)
{
    if (FoundOvershootOrDropOfVoltage(measurementObject) && numberOfMeasures!=0 )
    {
        numberOfOvershootMeasures++;
        UiManagerViewModel.Instance.OvershootDetected = UiManagerViewModel.
            Instance.OvershootDetected+
            "\n"+ "Overshoot was detected at: " + measurementObject.DateTimeValue.ToString() +
            " with Voltage="
            + measurementObject.VoltageValue.ToString()+ " V " ;
        UiManagerViewModel.Instance.Warnings = "Warnings: "+ "\n"+ "Overshoot was detected ";


    }
    else
    {
        SumPeaksOfVoltage += Math.Abs((double)measurementObject.VoltageValue);
        numberOfMeasures++;
        AveragePeakOfVoltage = SumPeaksOfVoltage / numberOfMeasures;
        UiManagerViewModel.Instance.AveragePeakVoltage = "AveragePeakVoltage: "
            + AveragePeakOfVoltage.ToString() + " V";


    }
```

Σχήμα 3.24: Μέθοδος που ενημερώνει το UI και εμφανίζει ένα μήνυμα εάν εντοπιστεί υπερταση ή πτώση τάσης

3.9.2 Συντελεστής Ισχύος Cp και Υπολογισμών

Ο τύπος για τον υπολογισμό του συντελεστή ισχύος Cp είναι:

$$Cp = \cos(\Delta\phi) \tag{3.5}$$

όπου Δφ είναι η διαφορά φάσης μεταξύ κυματομορφών τάσης και ρεύματος.

Δύο κύματα με την ίδια συχνότητα έχουν διαφορά φάσης εάν εμφανίζουν διαφορετικά πλάτη ταυτόχρονα ή στο ίδιο σημείο στο διάστημα. Η διαφορά φάσης εκφράζεται σε μοίρες ή ακτίνες και κυμαίνεται μεταξύ 0 και 360 ή 0 και 2π rad. [20]

Στο πλαίσιο αυτής της διπλωματικής, ο υπολογισμός της διαφοράς φάσης πραγματοποιήθηκε από το API. Κάθε μέτρηση ρεύματος ή τάσης αποθηκεύεται σε μια λίστα αντίστοιχα και τα σημεία που αντιπροσωπεύονται στα αντίστοιχα γραφήματα. Παρατηρείται ότι σύμφωνα με το ρυθμό μέτρησης, είναι βολικό να υποτεθεί ότι μια περίοδος περιλαμβάνεται σε εύρος ζώνης περίπου 12 σημείων.

Η κλάση CalculationsAndProcessOfChacrts (ελέγξτε το παράρτημα) διαχειρίζεται την εφαρμογή της διαφοράς φάσης. Αρχικά η μέθοδος getPointsOfPeriodInChart όπως φαίνεται στην παρακάτω εικόνα 3.25 επιλέγει (από τη λίστα όλων των μετρήσεων) τα σημεία που ανήκουν στην τρέχουσα περίοδο στην οποία θέλει να υπολογίσει τη διαφορά φάσης. Στη συνέχεια, οι μέθοδοι CalculateZeroPointOfCurrentInChart και CalculateZeroPointOfVoltageInChart υπολογίζουν τα σημεία όταν ο μηδενισμός του ρεύματος ή της τάσης εμφανίζονται αντίστοιχα. Από τα 2 σημεία που γίνεται ο μηδενισμός βρίσκεται ο αντίστοιχος ακριβής χρόνος τους (που πραγματοποιήθηκε ο μηδενισμός) με βάση αυτούς τος χρόνους γίνεται ο υπολογισμός της διαφοράς φάσης σύμφωνα με τον τύπο (3.6)

$$\Delta\varphi = \Delta t * 360 / T \qquad (3.6)$$

όπου Δφ η διαφορά φάσης, Δt η χρονική διαφορά μεταξύ των κυματομο ρφών του ρεύματος και της τάσης, T η περίοδος της κυματομορφής τάσης

```
2 references
public void getPointsOfPeriodInChart()
{
    currentCalculationList.Clear();
    if (ListOfMeasurmentsObject.Instance.getListOfMeasurementObject().Count <
        startingPoint + numberOfpointsOfAperiodInchart) { return; }

    for (int i = startingPoint; i < startingPoint+numberOfpointsOfAperiodInchart; i++)
    {
        currentCalculationList.Add(ListOfMeasurmentsObject.Instance.getMeasurementObjectFromList(i));
    }
    startingPoint = startingPoint + numberOfpointsOfAperiodInchart;
}
```

Σχήμα 3.25: Μέθοδος getPointsOfPeriodInChart.

Για τον υπολογισμό των μηδενικών σημείων ρεύματος και τάσης, οι μέθοδοι findMinValueOfVoltageInChart,findMaxValueOfVoltageInChart, findMinValueOfCurrentInChart, findMaxValueOfCurrentInChart εφαρμόζονται. Αυτές οι μέθοδοι όπως φαίνεται στην εικόνα 3.26, είναι υπεύθυνες για την εύρεση των ελάχιστων και μέγιστων σημείων που υπάρχουν σε κυματομορφές ρεύματος και τάσης

```
1 reference
public MeasurementObject findMinValueOfVoltageInChart()
{
    MeasurementObject minMeasurementObjectOfThisPeriod = currentCalculationList[0];

    foreach (MeasurementObject obj in currentCalculationList)
    {
        if (obj.VoltageValue < minMeasurementObjectOfThisPeriod.VoltageValue)
        {
            minMeasurementObjectOfThisPeriod = obj;
        }

    }
    return minMeasurementObjectOfThisPeriod;
}

2 references
public MeasurementObject findMaxValueOfCurrentInChart()
{
    MeasurementObject maxMeasurementObjectOfThisPeriod = currentCalculationList[0];
    foreach (MeasurementObject obj in currentCalculationList)
    {
        if (obj.CurrentValue > maxMeasurementObjectOfThisPeriod.CurrentValue)
        {
            maxMeasurementObjectOfThisPeriod = obj;
        }
    }

    return maxMeasurementObjectOfThisPeriod;
}
```

Σχήμα 3.26: Οι μέθοδοι findMaxValueOfVoltageInChart και findMinValueOfCurrentInChart.

Στη συνέχεια, η μέση τιμή μεταξύ αυτών των δύο σημείων υπολογίζεται χρησιμοποιώντας τη μέθοδο findMedianValue. Υπάρχουν πολλά βήματα που απαιτούνται για την εφαρμογή αυτών των υπολογισμών. Αρχικά, είναι απαραίτητο να επιλεχθεί ποια από τα δύο σημεία εμφανίζονται πρώτα στο χρόνο. Σε περίπτωση που το μέγιστο εμφανίζεται πρώτη φορά, διαγράφονται τα σημεία της λίστας που είναι χρονικά πριν από το μέγιστο και τα σημεία που είναι χρονικά μετά το ελάχιστο. Όταν ολοκληρωθεί η διαγραφή, συγκεντρώνονται εκείνα τα σημεία που είναι πλησιέστερα στο μηδέν σημείο και υπολογίζεται η μέση τιμή χρόνου . Με αυτόν τον τρόπο, αντιστοιχίζεται ένα σημείο για την τάση και ένα για το ρεύμα.

Η μέθοδος CalculateDeltaF όπως παρουσιάζεται στην εικόνα 3.28, εκτελεί τον τελικό υπολογισμό της διαφοράς φάσης. Αρχικά υπολογίζεται η διαφορά χρόνου μεταξύ τάσης και ρεύματος από τα 2 σημεία που βρέθηκαν. Στη συνέχεια χρησιμοποιείται η γραμμή κώδικα DeltaF = (360 * (Dt.TotalMilliseconds / 1000.0) * 50 * 0.01), που Dt.TotalMilliseconds είναι η διαφορά χρόνου.

```
1 reference
public void CalculateDeltaF(DateTime CurrentZeroTime)
{
    TimeSpan Dt = VoltageZeroTime.Subtract(CurrentZeroTime);
    //var microsec = Dt.TotalMilliseconds / (TimeSpan.TicksPerMillisecond / 1000);
    //var DeltaF = (360 * Dt.Seconds * 50 * 0.01);
    var DeltaF = (360 * (Dt.TotalMilliseconds / 1000.0) * 50 * 0.01);
    //var DeltaF = (360 * (Math.Abs(microsec) / 1000000.0) * 50 * 0.01);
    UiManagerViewModel.Instance.PowerCoefficient = "DeltaF: " + DeltaF.ToString() + " degrees";
}
```

Σχήμα 3.28: Μέθοδος CalculateDeltaF .

4 Συμπεράσματα

Συμπερασματικά για μελλοντική ανάπτυξη του μοντέλου θα μπορούσε να προσαρμοστεί η συσκευή στον ηλεκτρικό πίνακα ενός σπιτιού. Με αυτόν τον τρόπο θα είναι δυνατή η παρακολούθηση της συνολικής κατανάλωσης του σπιτιού. Συνδέοντας έναν μετατροπέα AC-AC που μπορεί να επεξεργαστεί ένα συγκεκριμένο εύρος τάσης και εύρος ζώνης συχνότητας, η συσκευή μπορεί να μετρήσει και τα δύο και ακόμη και να καταγράψει την παραμόρφωση.

Μια άλλη πιθανή επέκταση είναι η διαχείριση του προγράμματος από ένα κινητό τηλέφωνο.

5 Βιβλιογραφία

[1]. «Smart Grid: Technology and Applications», Janaka B. Ekanayake, Nick Jenkins, Kithsiri Liyanage, Jianzhong Wu, Akihiko Yokoyama

[2]. «Fully Integrated, Hall Effect-Based Linear Current Sensor IC and a Low-Resistance Current Conductor,» Allegro MicroSystems, LLC, Worcester, Massachusetts, 2015

[3]. https://www.elecrow.com/acs712-current-sensor-30a-p-710.html

[4]. «ZMPT101B Current-type Voltage Transformer», Q. Z. L. Electronic, May 2015

[5]. «ZMPT101B(ZMPT107) voltage transformer operating guide», NanJing ZeMing Electronic Co., Ltd, 2014

[6]. http://ldsrc.blogspot.com/2018/02/micro-sd-card-for-esp32.html

[7]. https://grobotronics.com/esp32-development-board-esp32-devkitc-32d.html?sl=en

[8]. «Wireless Networking: Introduction to Bluetooth and WiFi», Gordon Colbach

[9]. «802.11 WiFi Standards Explained», Mitchell, Bradley (3 October 2018). Lifewire. Archived from the original on 12 December 2018. Retrieved 18 August 2018.

[10]. «Δίκτυα Υπολογιστών», Andrew S. Tanenbaum, 4η Αμερικανική Έκδοση

[11]. https://www.geeksforgeeks.org/introduction-to-visual-studio/

[12]. https://docs.microsoft.com/en-us/visualstudio/debugger/debugging-absolute-beginners?view=vs-2019

[13]. «Modern Programming Languages: A Practical Introduction», Adam Brooks Webber, 2003

[14]. https://www.c-sharpcorner.com/article/what-is-c-sharp/

[15]. https://www.wpf-tutorial.com/about-wpf/what-is-wpf/

[16].» Database System Concepts», Avi Silberschatz, Henry F. Korth, S. Sudarshan, 6th Edition

[17]. https://www.ionos.com/digitalguide/server/tools/xampp-tutorial-create-your-own-local-test-server/

[18]. https://www.awardspace.com/kb/database-connection-settings/

[19]. https://www.researchgate.net/figure/ACS712-step-down-resistors-connection_fig3_334661815

[20]. «Εισαγωγή στα Συστήματα Ηλεκτρικής Ενέργειας», Βουρνάς Κ., Κονταξής Γ.

6 Παράρτημα

Ο κώδικας που υλοποιεί την κλάση

CalculateZeroPointOfCurrentInChart είναι:

```
public DateTime CalculateZeroPointOfCurrentInChart()

{

double medianValue =
findMedianValue((double)findMinValueOfCurrentInChart().CurrentValue,

(double)findMaxValueOfCurrentInChart().CurrentValue);

int Minindex = currentCalculationList.FindIndex(a => a ==
findMinValueOfCurrentInChart());

int Maxindex = currentCalculationList.FindIndex(b => b ==
findMaxValueOfCurrentInChart());

//remove from list left and right side of object that not needed to be calculated

//removal depends on the position of the indexes of min and max values

if (Minindex < Maxindex)

{

MeasurementObject leftSideObj = null;

if (!isMinFirst)

{

//an einai panw apo to 0 to prwto stoixeio ths listas kopse thn aristeri pleura alliws pare
apo thn arxh ths listas

if (currentCalculationList[0].CurrentValue > 0M)
```

```
{
currentCalculationList.RemoveRange(0, Maxindex);
leftSideObj = currentCalculationList[0];
foreach (MeasurementObject obj in currentCalculationList)
{
double diff = medianValue - (double)obj.CurrentValue;
if (diff < 0 && (double)leftSideObj.CurrentValue > (double)obj.CurrentValue)
{ leftSideObj = obj; } } }
Else {
currentCalculationList.RemoveRange(Minindex, currentCalculationList.Count - Minindex
- 1);
leftSideObj = currentCalculationList[0];
foreach (MeasurementObject obj in currentCalculationList) {
double diff = medianValue - (double)obj.CurrentValue;
if (diff < 0 && (double)leftSideObj.CurrentValue > (double)obj.CurrentValue) {
leftSideObj = obj; } } } }
else{
if (Minindex == 0) { currentCalculationList.RemoveAt(0); }
else{
currentCalculationList.RemoveRange(0, Minindex);}
int newMaxIndex = Maxindex - Minindex;
```

```
//currentCalculationList.RemoveRange(newMaxIndex+1, currentCalculationList.Count -
newMaxIndex);

currentCalculationList.RemoveRange(newMaxIndex, currentCalculationList.Count -
newMaxIndex);

leftSideObj = currentCalculationList[0];

foreach (MeasurementObject obj in currentCalculationList)

{

double diff = Math.Abs(medianValue) - (double)obj.CurrentValue;

if (diff > 0 && (double)leftSideObj.CurrentValue < (double)obj.CurrentValue)

{

leftSideObj = obj;

}

}

}

int Leftindex = currentCalculationList.IndexOf(leftSideObj);

MeasurementObject rightSideObj = null;

if (Leftindex == currentCalculationList.Count - 1 || Leftindex == 0)

{

rightSideObj = currentCalculationList[Leftindex];

}

else
```

```
{
rightSideObj = currentCalculationList[Leftindex + 1];
}
TimeSpan TimeDiff = leftSideObj.DateTimeValue - rightSideObj.DateTimeValue;
TimeSpan ZeroValueDateTime = new TimeSpan(TimeDiff.Ticks / 2);
// System.Console.WriteLine("Zero Current Time is: " + ZeroValueDateTime);
CurrentZeroTime = rightSideObj.DateTimeValue.Add(ZeroValueDateTime);
}
else if (Minindex > Maxindex)
{
MeasurementObject leftSideObj = null;
if (isMinFirst)
{
//an einai panw apo to 0 to prwto stoixeio ths listas kopse thn aristeri pleura alliws pare apo thn arxh ths listas
if (currentCalculationList[0].CurrentValue > 0M)
{
currentCalculationList.RemoveRange(0, Minindex);
leftSideObj = currentCalculationList[0];
foreach (MeasurementObject obj in currentCalculationList)
{
```

```
double diff = medianValue - (double)obj.CurrentValue;
if (diff > 0 && (double)leftSideObj.CurrentValue < (double)obj.CurrentValue)
{
leftSideObj = obj;
}
}
}
else
{
currentCalculationList.RemoveRange(Maxindex, currentCalculationList.Count -
Maxindex - 1);
leftSideObj = currentCalculationList[0];
foreach (MeasurementObject obj in currentCalculationList)
{
double diff = medianValue - (double)obj.CurrentValue;
if (diff > 0 && (double)leftSideObj.CurrentValue < (double)obj.CurrentValue)
{
leftSideObj = obj;
}
}
}
```

```
}
else
{
currentCalculationList.RemoveRange(0, Maxindex);
int newMinIndex = Minindex - Maxindex + 1;
currentCalculationList.RemoveRange(newMinIndex, currentCalculationList.Count -
newMinIndex);
leftSideObj = currentCalculationList[0];
foreach (MeasurementObject obj in currentCalculationList)
{
double diff = medianValue - (double)obj.CurrentValue;
if (diff < 0 && (double)leftSideObj.CurrentValue > (double)obj.CurrentValue)
{
leftSideObj = obj;
}
}
}
int Leftindex = currentCalculationList.IndexOf(leftSideObj);
MeasurementObject rightSideObj = null;
if (Leftindex == 0 || Leftindex == currentCalculationList.Count-1)
{
```

```
rightSideObj = currentCalculationList[Leftindex];

}

else

{

rightSideObj = currentCalculationList[Leftindex + 1];

}

TimeSpan TimeDiff = rightSideObj.DateTimeValue - leftSideObj.DateTimeValue;

TimeSpan ZeroValueDateTime = new TimeSpan(TimeDiff.Ticks / 2);

CurrentZeroTime = leftSideObj.DateTimeValue.Add(ZeroValueDateTime);

}

return CurrentZeroTime;

}
```

Κώδικας σε .ino για την σειριακή λειτουργία του Esp32:

```
#include "ZMPT101B.h"

#include "ACS712.h"

const int currentPin = A0;

const int voltagePin = A1;

int voltageInput =0;

int currentInput=0;

float current=0;

ZMPT101B voltageSensor(voltagePin);

ACS712 currentSensor(ACS712_05B, currentPin);

void setup()

{

Serial.begin(9600);

}

void loop() {

while (Serial.available() == 0)

{

MeasureUIP();

}
```

```
}

void MeasureUIP()

{

voltageInput  = analogRead(voltagePin);

float voltage = (voltageInput - 508) * 2.5; //callibrated to be around 0 (508) and reverse ref = 2.5

currentInput  = analogRead(currentPin);

current = ((510 - currentInput )/1024.0 ) * 250  * 1000/10; //calibrated

float U = (voltageSensor.getVoltageAC()-1) *10.5; //calibration

float I = (currentSensor.getCurrentAC()*1000 - 45.76) *3.28; //in mA with calibration

float P = U * I / 1000;

Serial.println(String("U=") + voltage  +"I="+ current +"P=" + P );

Serial.println(String("Vr=") + U);  //Calibrated

Serial.println(String("Ir=") + I);  //Calibrated

 delay(100);

}
```

Κώδικας σε .ino για την Wifi λειτουργία του Esp32:

```
#include <WiFi.h>

#include <DNSServer.h>

#include <WebServer.h>

#include <WiFiManager.h>

#include "ZMPT101B.h"

#include "ACS712.h"

const char* host = "192.168.1.4"; //IP syndesh me alla diktya. take it from ipconfig in cmd where Lan IPv4Address

const uint16_t port = 8022;

WebServer server(80);

WiFiManager wifiManager;

const int currentPin = 39; //it worked here when connected to 33??

const int voltagePin = 36;

int voltageInput =0;

int currentInput=0;

float current=0;

ZMPT101B voltageSensor(voltagePin);

ACS712 currentSensor(ACS712_05B, currentPin);

WiFiClient client;

int isconnected=0;
```

```
void setup() {

delay(1000);

Serial.begin(115200);

Serial.println();

wifiManager.setAPCallback(configModeCallback);

wifiManager.setTimeout(60);

if(!wifiManager.autoConnect("ESP32"))

{

Serial.println("Failed to connect to router and hit timeout");

}else

{

Serial.println("Successfully connected to router");

}

server.begin();

Serial.print("IP Adress: ");

Serial.println(WiFi.localIP());

}

void loop() {

if(isconnected==0)

{
```

```
if (!client.connect(host, port))

{

Serial.println("connection to API's Server failed");

Serial.println("wait 5 sec...");

delay(5000);

return;

}else

{

Serial.println("connection to API's Server succeeded");

isconnected=1;

}

}

MeasureUIP(client);

}

void configModeCallback(WiFiManager *myWiFiManager)

{

Serial.println("Entered config mode");

Serial.println(WiFi.softAPIP());

Serial.println(myWiFiManager->getConfigPortalSSID());

}
```

```
void MeasureUIP(WiFiClient cl)

{  voltageInput  = analogRead(voltagePin);

float voltage = voltageInput * (3.3 / 4096.0) * 100; //gia ESP32 ????

currentInput  = analogRead(currentPin);

current = (((currentInput-512) / 4096.0 )* 3300) *1000/100;

float U = voltageSensor.getVoltageAC() * 10;

float I = currentSensor.getCurrentAC() * 1000;

float P = U * I / 1000;

cl.println(String("U=") + voltage  +"I="+ current +"P=" + P );

delay(100);

}
```