ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ


# ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΕΠΙΤΗΡΗΣΗΣ ΚΑΙ ΚΑΤΑΓΡΑΦΗΣ ΗΛΕΚΤΡΙΚΗΣ ΕΝΕΡΓΕΙΑΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΗ ΕΝΕΡΓΕΙΑΚΗ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΣΕ «ΕΥΦΥΕΙΣ ΚΑΤΟΙΚΙΕΣ»

Διπλωματική Εργασία

της

Ελένης Αντωνάκη

Θεσσαλονίκη, Οκτώβριος 2018

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΕΠΙΤΗΡΗΣΗΣ ΚΑΙ ΚΑΤΑΓΡΑΦΗΣ ΗΛΕΚΤΡΙΚΗΣ ΕΝΕΡΓΕΙΑΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΗ ΕΝΕΡΓΕΙΑΚΗ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΣΕ «ΕΥΦΥΕΙΣ ΚΑΤΟΙΚΙΕΣ»

Ελένη Αντωνάκη

Πτυχίο Μηχανικών Πληροφορικής Τ.Ε., Αλεξάνδρειο ΤΕΙ Θεσσαλονίκης, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Ψάννης Κωνσταντίνος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6/11/2018

| Ψάννης Κων/νος | Μαργαρίτης Κων/νος | Ευαγγελίδης Γεώργιος |
|---|---|---|
| ..... | ..... | ..... |

Αντωνάκη Ελένη

.....

**Ευχαριστίες**

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή μου κ. Κωνσταντίνο Ψάννη για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου το συγκεκριμένο θέμα, την άριστη συνεργασία και τη συνεχή υποστήριξή του καθ' όλη την πορεία της εργασίας αυτής. Επίσης θα ήθελα να ευχαριστήσω τις τεχνολογικές κοινότητες SKGTech για τις ανταλλαγές απόψεων και το ειλικρινές ενδιαφέρον τους.

Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου και στους αφανείς ήρωες αυτής της προσπάθειας.

# Περίληψη

Η εξοικονόμηση ενέργειας και η ανάπτυξη μεθόδων για την αποδοτική χρήση της αποτελεί εδώ και χρόνια αντικείμενο μελέτης μεγάλου μέρους της ερευνητικής κοινότητας. Ένας από τους σημαντικότερους τομείς κατανάλωσης ενέργειας είναι ο κτηριακός με ιδιαίτερη εστίαση στις κατοικίες. Η ραγδαία τεχνολογική εξέλιξη τα τελευταία χρόνια στο χώρο του Διαδικτύου των Αντικειμένων (Internet of Things - IoT) με την εξέλιξη της δικτύωσης και της αλληλεπίδρασης των διάφορων συσκευών και συστημάτων, καθιστά εφικτή την ανάπτυξη εφαρμογών, στον τομέα των «Έξυπνων Οικιών», που δύναται να εξυπηρετήσει σε μεγάλο βαθμό τη διαχείριση ενέργειας σε οικιακές εγκαταστάσεις.

Στην παρούσα έρευνα σχεδιάστηκε ένα σύστημα που υλοποιεί την καταγραφή των ενεργειακών δεδομένων, έχοντας ως στόχο την ανάπτυξη μεθόδων για τη βελτίωση των ενεργειακών συνηθειών των καταναλωτών και τη μείωση της άσκοπης κατανάλωσης ηλεκτρικής ισχύος. Η υλοποίηση της ιδέας βασίστηκε στην κατασκευή έξυπνων ρευματοδοτών και ενός αξιόπιστου συστήματος για την απομακρυσμένη διαχείριση των φορτίων και την καταγραφή των μετρήσεων αυτών. Επιπλέον αναπτύχθηκε μια διαδικτυακή εφαρμογή διαχείρισης που παρέχει στον καταναλωτή τη δυνατότητα ελέγχου των συσκευών σε πραγματικό χρόνο και δημιουργίας σεναρίων χρονοπρογραμματισμού. Οι χρήστες μπορούν να λαμβάνουν στατιστικά δεδομένα και να έχουν μια εικόνα του ενεργειακού προφίλ της οικίας τους, έτσι ώστε οι ίδιοι, μέσω του ενεργού τους ρόλου, να λαμβάνουν βέλτιστες αποφάσεις.

**Λέξεις Κλειδιά:** Διαδίκτυο των Αντικειμένων, Ενεργειακή Διαχείριση, Απομακρυσμένος έλεγχος ρεύματος, Arduino, Ηλεκτρονόμος, Αισθητήρας έντασης ρεύματος, MQTT, Προγραμματισμός διαδικτύου, ASP.NET, MVC.

# Abstract

Energy saving and the development of methods for its efficient use, has been the main object of the majority in the research community, for many years. One of the most important sectors of energy consumption is the residential with a particular focus on housing. Over the last few years, the rapid technological advancement in the field of Internet of Things (IoT), with the evolution of the networking and the interaction of various devices and systems, makes the development possible for applications in the section of "Smart Homes", which can greatly serve the management of energy in residential facilities.

In the present study, a system was developed that implements the recording of the energy data, aiming at the development of methods for the improving of consumers' energy, habits and reducing the unnecessary power consumption. The implementation of such an idea was based on the construction of smart plugs and of a reliable system for the remote handling of loads and the recording of these measurements. In addition, an online management application has been developed that provides to the consumer the ability to control devices in real-time and create scheduling scenarios. Users can receive statistics and have a view of their home's energy profile so that they, through their active role, make optimal decisions.



**Keywords:** Internet of Things, Energy saving, Remote Power Control, Arduino, Relay, Current Sensor, MQTT, Internet programming, ASP.NET, MVC

# Περιεχόμενα

**Ευρετήριο Εικόνων**

**Ευρετήριο Πινάκων**

# 1 Εισαγωγή

Τα τελευταία χρόνια υπάρχει μια διαρκής και ολοένα αυξανόμενη τάση της διασύνδεσης ποικίλων συσκευών ή εξαρτημάτων με το διαδίκτυο. Η διασύνδεση αυτή προσδίδει νέα χαρακτηριστικά και τρόπους χρήσης, ενισχύοντας τη λειτουργικότητα και τον έλεγχο των διασυνδεδεμένων συσκευών.

Ως συνέπεια της έντονης αυτής τάσης οδηγηθήκαμε στη γέννηση του όρου "Διαδίκτυο Αντικειμένων" (Internet of things), κατά τον οποίο κάθε συσκευή μπορεί να ελεγχθεί ή να ερωτηθεί για την κατάστασή της μέσω του διαδικτύου.

## 1.1 Αντικείμενο Διπλωματικής Εργασίας

Η εξοικονόμηση ενέργειας και η ανάπτυξη μεθόδων για την αποδοτική χρήση της αποτελεί στόχο της ενεργειακής πολιτικής καθώς και αντικείμενο μελέτης μεγάλου μέρους της ερευνητικής κοινότητας. Ένας από τα σημαντικούς τομείς κατανάλωσης ενέργειας είναι ο κτηριακός τομέας, με εστίαση στις κατοικίες.

Η υπερκατανάλωση της ενέργειας σε μία οικία οφείλεται σε σημαντικό βαθμό στην άγνοια του τελικού καταναλωτή, λόγω ελλιπούς πληροφόρησης, όσον αφορά την ενέργεια που καταναλώνεται και πώς αυτή συνδέεται με τις διάφορες δραστηριότητές του. Η παρακολούθηση και καταγραφή της ενεργειακής διακύμανσης κρίνεται απαραίτητη, αν λάβει κανείς υπόψη και το γεγονός ότι πολλές φορές οι διαθέσιμοι ενεργειακοί πόροι είναι πεπερασμένοι.

Με την σύγχρονη ραγδαία ανάπτυξη της τεχνολογίας και του Internet Of Things (IoT) έχει δημιουργηθεί μια πρωτοφανής παραγωγή ποικίλων δεδομένων προερχόμενες από «έξυπνες» συσκευές που καλούνται οι νέες τεχνολογικές λύσεις να την αξιοποιήσουν. Με τη συστηματική και αυτοματοποιημένη συγκέντρωση και την κατάλληλη επεξεργασία των δεδομένων μπορεί να επιτευχθεί η αποδοτική διαχείριση της ενέργειας.

Στο πλαίσιο αυτό η παρούσα διπλωματική εργασία υλοποιεί ένα σύστημα, το οποίο εντάσσεται στον τομέα των «Έξυπνων Οικιών» και πραγματεύεται τον απομακρυσμένο έλεγχο ηλεκτρικής ισχύος. Η διερεύνηση του χώρου των τεχνολογιών, των πρωτοκόλλων και των αρχιτεκτονικών υλοποίησης συστημάτων

του Διαδικτύου των Αντικειμένων αποτελεί τη βασική θεματογραφία που διαμόρφωσε και τον άξονα της μελέτης μας.

Η ιδέα αυτού του συστήματος, θα μπορούσε να εφαρμοστεί σε ένα μεγάλο εύρος εφαρμογών, ακόμη και σε επαγγελματικές ενεργές δικτυακές υποδομές (datacenters). Αν και υπάρχουν συσκευές που μπορεί να χρησιμοποιηθούν για τον έλεγχο της ενέργειας μέσω διαδικτύου, παρουσιάζουν κάποια μειονεκτήματα, με το μεγάλο κόστος να αποτελεί ένα από τα πιο σημαντικά. Το παρόν σύστημα καλείται να ξεπεράσει αυτό το μειονέκτημα παρέχοντας μια οικονομική και αποτελεσματική εναλλακτική στις ήδη υπάρχουσες υλοποιήσεις, εστιάζοντας στη χρήση των προσωπικών οικιακών συσκευών.

Προσφέρει μια πληθώρα επιπρόσθετων χαρακτηριστικών, όπως τη δυνατότητα χρονοπρογραμματισμού ενεργειών, τη δυνατότητα παροχής στοιχείων σχετικά με την κατανάλωση ρεύματος, και ενημέρωσης του χρήστη για την κατάσταση των συσκευών του σε πραγματικό χρόνο. Αξιοσημείωτο χαρακτηριστικό αποτελεί και η δυνατότητα εύκολης ενσωμάτωσης λειτουργιών του συστήματος σε υλοποιήσεις τρίτων.

## 1.2 Σύνοψη Διπλωματικής Εργασίας

Στα κεφάλαια που ακολουθούν γίνεται αναλυτική παρουσίαση των πειραματικών διατάξεων με τις τεχνολογίες και τις αρχιτεκτονικές προσεγγίσεις που εφαρμόστηκαν. Αναλύονται οι λειτουργίες που μπορεί να πραγματοποιήσει το τελικό σύστημα, αιτιολογούνται επιλογές που ακολουθήθηκαν στη σχεδίαση του υλικού και του λογισμικού, και παρουσιάζονται οι τρόποι με τους οποίους ο χρήστης μπορεί να αλληλοεπιδρά με το σύστημα.

Το κείμενο δομείται σε επτά κεφάλαια, κάθε ένα από τα οποία άπτεται ζητήματα τα οποία αφορούν την υλοποίηση που ακολουθήθηκε στο σύνολό της.

- → Στο παρόν κεφάλαιο γίνεται η παρουσίαση αντικειμένου που πραγματεύεται η παρούσα διπλωματική εργασία, καθώς και μια σύνοψη της δομής της.
- → Στο δεύτερο κεφάλαιο, αναλύεται το θεωρητικό υπόβαθρο από το οποίο αντλήθηκαν οι γνώσεις.
- → Στο τρίτο κεφάλαιο μελετώνται αυτόματα συστήματα και υλοποιήσεις σε μία Έξυπνη Οικία που εστιάζουν στην ενεργειακή διαχείριση και την

αυτοματοποίηση και αναλύεται η μεθοδολογία ανάπτυξης του συστήματος της παρούσας εργασίας.

- → Στο τέταρτο κεφάλαιο παρουσιάζονται οι γλώσσες προγραμματισμού, τα εργαλεία και οι πλατφόρμες ανάπτυξης που χρησιμοποιήθηκαν καθώς και τα τεχνικά χαρακτηριστικά μετασχηματιστών ρεύματος και μικροελεγκτών, που αποτέλεσαν και τη βάση της υλοποίησης.
- → Στο πέμπτο κεφάλαιο, παρουσιάζονται τα στάδια σχεδίασης και υλοποίησης των πειραματικών διατάξεων που επιχειρήθηκαν, με αναλυτικά σχηματικά των κυκλωμάτων και των συνδεσμολογιών. Ακόμη παρατίθεται ο προγραμματισμός του μικροελεγκτή για κάθε σενάριο με τις απαραίτητες επεξηγήσεις.
- → Στο έκτο κεφάλαιο, επεξηγείται η αρχιτεκτονική του τελικού συστήματος καθώς και η ανάπτυξη της διαδικτυακής εφαρμογής. Συγκεκριμένα, αναλύεται ένα κομμάτι του σχεδιασμού της βάσης δεδομένων, η εφαρμογή συστήματος και το πρότυπο σχεδίασης της με παραρτήματα κώδικα, δίνοντας ιδιαίτερη έμφαση στην υλοποίηση της καταγραφής των μετρήσεων.
- → Στο έβδομο και τελευταίο κεφάλαιο, συνοψίζεται το έργο της διπλωματικής εργασίας, τα συμπεράσματα που ανακύπτουν. Επίσης παρουσιάζονται τα σημεία στα οποία αυτή ενδεχομένως πλεονεκτεί απέναντι σε αντίστοιχες εμπορικές υλοποιήσεις, καθώς και οι μελλοντικές επεκτάσεις που μπορούν να προστεθούν στο σύστημα ώστε να το αναβαθμίσουν.

# 2 Θεωρητικό Υπόβαθρο

Στόχος του κεφαλαίου αυτού είναι η θεωρητική προσέγγιση του συνόλου των στοιχείων που απαρτίζουν την παρούσα διπλωματική εργασία. Αποσαφηνίζονται θεωρητικές έννοιες και ορολογίες που αφορούν τις τεχνολογίες του διαδικτύου και την επιστήμη της ηλεκτρονικής.

## 2.1 Το Διαδίκτυο των Αντικειμένων

Ο όρος «Διαδίκτυο των Αντικειμένων» (IoT) χρησιμοποιήθηκε για πρώτη φορά το 1999 από τον Βρετανό πρωτοπόρο στην τεχνολογία, Kevin Ashton, για να περιγράψει ένα σύστημα στο οποίο τα αντικείμενα του φυσικού κόσμου θα μπορούσαν να συνδεθούν με το διαδίκτυο [1]. Ο Ashon αποσκοπούσε στο να τονίσει τη δύναμη των συστημάτων ταυτοποίησης μέσω ραδιοσυχνοτήτων, γνωστά ως RFID (Radio Frequency Identification) [2], που χρησιμοποιούνται από εταιρικές εφοδιαστικές αλυσίδες, προκειμένου να μετρήσουν και να παρακολουθούν τα εμπορεύματα χωρίς την ανάγκη για ανθρώπινη παρέμβαση.

Σήμερα, το Διαδίκτυο των Αντικειμένων έχει γίνει ένας δημοφιλής όρος για την περιγραφή σεναρίων στα οποία η σύνδεση στο Internet και οι δυνατότητες των υπολογιστών επεκτείνονται στη σύσταση ενός δικτύου από φυσικά αντικείμενα, συσκευές, αισθητήρες και γενικότερα οποιαδήποτε αντικείμενο του τεχνητού ανθρώπινου περιβάλλοντος. Κάθε ένα από αυτά τα αντικείμενα περιέχει ενσωματωμένη τεχνολογία έτσι ώστε να μπορεί να αλληλοεπιδρά με εσωτερικές καταστάσεις ή το εξωτερικό περιβάλλον, να συλλέγει και να ανταλλάσσει δεδομένα. Η σύνδεση με τοπικό δίκτυο ή το διαδίκτυο, επιτρέπει στον χρήστη την παρακολούθηση αισθητήρων καθώς και τον απομακρυσμένο έλεγχο της συσκευής.

Το Διαδίκτυο των Αντικειμένων μπορεί να θεωρηθεί ως ιδέα προς την ύψιστη αξιοποίηση και ανάπτυξη των διασυνδεδεμένων τεχνολογιών και ειδικότερα του διαδικτύου. Αποτελεί την όλο και μεγαλύτερη είσοδο και ύφανση των μικροϋπολογιστών και των τηλεπικοινωνιών, στο ανθρώπινο και φυσικό περιβάλλον.

### 2.1.1 Machine to Machine Communication (M2M)

Ο όρος Machine to Machine (M2M) Communication αναφέρεται στην απευθείας επικοινωνία μεταξύ συσκευών μέσω οποιουδήποτε καναλιού επικοινωνίας (πχ. Ενσύρματο ή ασύρματο). Η επικοινωνία M2M είναι στενά συνδεδεμένη με το Διαδίκτυο των Αντικειμένων, καθώς συσκευές καλούνται να επικοινωνήσουν μεταξύ τους για την αυτόματη ανταλλαγή δεδομένων. Η υποστήριξη του M2M βασίζεται σε υλοποιήσεις που έχουν αναπτύξει διάφοροι οργανισμοί. Πιο γνωστοί οργανισμοί είναι το Eclipse machine to machine working group, η OASIS MQTT, η XMPP και η Open Mobile Alliance [3].

### 2.1.2 Ενδεικτικά πρωτόκολλα και αρχιτεκτονικές

Απαραίτητη προϋπόθεση για την εγκαθίδρυση ενός δικτύου Internet of Things είναι η προσπάθεια μιας ολοκληρωμένης ενοποίησης της πληθώρας των δικτύων που εσωκλείονται και των διαφορετικών πρωτοκόλλων επικοινωνίας που εφαρμόζονται σε κάθε επίπεδο. Τα επίπεδα αυτά, που ορίζονται για την καλύτερη σχεδίαση του δικτύου, μπορούν να συμπεριληφθούν στις εξής βασικές κατηγορίες:

- Φυσικό επίπεδο μεταφοράς δεδομένων
- Πρωτόκολλο μεταφοράς δεδομένων
- Πρωτόκολλο λειτουργίας (IoT) δικτύου

Η πρώτη κατηγορία ανήκει στο χαμηλότερο επίπεδο της στοίβας του δικτύου. Καθορίζει το φυσικό μέσο μετάδοσης και τη διαμόρφωση της πληροφορίας πάνω σε αυτό. Ιδιαίτερο ενδιαφέρον στο διαδίκτυο των αντικειμένων παρουσιάζουν οι ασύρματες τεχνολογίες [4], δεδομένου ότι η πλειονότητα των εφαρμογών βασίζεται στην ασύρματη επικοινωνία των συσκευών.

Δημοφιλέστερα είναι τα LPWANs (Low Power Wide Area Networks). Αυτά τα δίκτυα επιτρέπουν ασύρματη επικοινωνία σε ευρείες, ακόμα και εθνικές, εκτάσεις, με χρήση της ελάχιστης δυνατής ενέργειας από τις συσκευές. Υπάρχουν δύο κατηγορίες τέτοιων δικτύων: τα Cellular με τα οποία αναφερόμαστε στις τεχνολογίες που έχουν προκύψει από την κινητή τηλεφωνία (LTE Cat-M1, NB-IoT/Cat-M2, EC-GSM, 5G) και τα Non-Cellular (LoRa, SigFox).

Για ασύρματη επικοινωνία σε μικρότερους χώρους υπάρχουν τα WPANs και WLANS (Wireless Personal/Local Area Networks) [5]. Η μικρή τους εμβέλεια, τα

καθιστά χρήσιμα κυρίως σε περιπτώσεις αυτοματισμών οικιών, γραφείων και γενικότερα κτηρίων. Αυτό φαίνεται και από τις εταιρείες, οι οποίες δηλώνουν ενδιαφέρον για τις τεχνολογίες αυτές.

- Wi-Fi: Το Wireless Fidelity (WiFi) είναι ένας κοινός όρος που αναφέρεται στο πρότυπο της IEEE, IEEE 802.11.x, το οποίο είναι κατάλληλο για τη δημιουργία ασύρματων τοπικών δικτύων (WLANS). Υποστηρίζεται από το WiFi Alliance, ένα μη κερδοσκοπικό οργανισμό με πάνω από 300 εταιρείες.
- WiFi HaLow (Low Power): Πρόκειται για ένα νέο ασύρματο πρότυπο IEEE 802.11ah. Η WiFi Alliance ανακοίνωσε το WiFi HaLow τον Ιανουάριο του 2016.
- Bluetooth Low Energy (BLE): Ονομάζεται αλλιώς Bluetooth 4.0 ή Bluetooth Smart και αποτελεί την εξέλιξη της καθολικής τεχνολογίας ασύρματης διεπαφής Bluetooth. Το κύριο πλεονέκτημα του είναι η μειωμένη καταναλισκόμενη ισχύς σε σύγκριση με τους προκατόχους του ενώ παράλληλα μπορεί να εκπέμψει στην ίδια απόσταση με αυτούς διατηρώντας το μειωμένο κόστος παραγωγής.
- ZigBee: Αποτελεί μια προδιαγραφή για μια σουίτα πρωτοκόλλων επικοινωνίας υψηλού επιπέδου σύμφωνα με το πρότυπο IEEE 802.15.4, ένα πρότυπο χαμηλής ισχύος για ασύρματα δίκτυα προσωπικής περιοχής (WPANs). Υποστηρίζεται από το ZigBee Alliance και είναι τεχνολογία κατάλληλη για εφαρμογές που απαιτούν χαμηλή κατανάλωση ενέργειας και χαρακτηρίζονται από χαμηλό ρυθμό μετάδοσης δεδομένων.
- Z-Wave: Είναι ένα ιδιόκτητο ασύρματο πρότυπο επικοινωνιών ειδικά σχεδιασμένο για εφαρμογές απομακρυσμένου ελέγχου σε οικιακό και ελαφρά επαγγελματικό περιβάλλον. Υποστηρίζεται από το Z-Wave Alliance.

**Πίνακας 2-1: Σύγκριση Τεχνολογιών Ασύρματης Επικοινωνίας**

| | | ZigBee | WiFi (Low Power) | Bluetooth | BLE | Z-Wave |
|---|---|---|---|---|---|---|
| Standardization | | IEEE 802.15.4 | IEEE 802.11.ah | IEEE 802.15.1 | IEEE 802.15.1 | Proprietary |
| Frequency | | 2.4 GHz, 868, 915 MHz | 900 MHz | 2.4 GHz | 2.4 GHz | 900 MHz |
| Range, m | indoor | 10-100 | < 700 | 1, 10, 100 | 50 | 30 |
| | outdoor | | < 1000 | | | |
| Data rate | | 20, 40, 250 Kb/s | 150-400, 650-780 Kb/s | 1, 2, 3 Mb/s | 1Mb/s | 9.6, 40, 100 Kb/s |
| Throughtput | | 10 - 115.2 Kb/s | > 100 Kb/s | 0.7-2.1 Mb/s | 305 Kb/s | - |
| Power consumption, mA | | < 40 | - | < 30 | < 12.5 | < 23 |
| TX output power, dBm | from | -3 | 10 | -6 | < 19 | < 0 |
| | to | 10 | 30 | 20 | | |
| Security Algorithm | | AES-128 | WEP, WPA, WPA2 | E0, E1, E21, E22, E3, 56-128 bit | AES-128 | AES-128 |
| Topology | | star, tree, mesh | one-hop | p2p scatternet | p2p, star | star, mesh |

Πηγή: http://www.diva-portal.org/smash/get/diva2:1118965/FULLTEXT02 [6]

Το πρωτόκολλο μεταφοράς δεδομένων αποτελεί τη δεύτερη βασική κατηγορία και ανήκει στο ενδιάμεσο επίπεδο της στοίβας ενός δικτύου IoT, ορίζει τους κανόνες, τις διαδικασίες και τους τρόπους, με τους οποίους ορίζεται και λειτουργεί αυτό το δίκτυο. Σε αυτό το επίπεδο θεωρείται δεδομένη η χρήση του πρωτοκόλλου IP (Internet Protocol). Το πρωτόκολλο αυτό αποτελεί την βάση του σημερινού

διαδικτύου και την κυρίαρχη διασυνδεδεμένη τεχνολογία. Η 6η έκδοσή του (IPv6), θα μπορεί να υποστηρίζει ταυτόχρονα, $2^{128}$ δηλαδή 3.4 x $10^{38}$ διαφορετικές διευθύνσεις και κατά συνέπεια συσκευές [7].

Η τρίτη κατηγόρια ανήκει στο υψηλότερο επίπεδο της στοίβας του δικτύου όπου η επικοινωνία αποτελεί πλέον μεταφορά δεδομένων μεταξύ IoT κόμβων/συσκευών. Σε αυτήν την κατηγορία εντάσσεται το πρωτόκολλο επικοινωνίας που ορίζει τους κανόνες με τους οποίους λειτουργεί το IoT δίκτυο. Με την εξέλιξη των υπαρχόντων πρωτοκόλλων και την ανάπτυξη νέων καλείται η επίλυση διάφορων ζητημάτων σχετικά με τη λειτουργικότητα, την επίδοση, την ασφάλεια κ.α. Παρακάτω παρουσιάζονται ενδεικτικά μερικά από αυτά που έχουν απήχηση στον τομέα του Internet of Things.

**HTTP (HyperText Transfer Protocol)**

Αρκετές επιστημονικές δημοσιεύσεις προτείνουν τη χρήση του HTTP για την ομαλή ανάπτυξη του IoT (και κυρίως της M2M επικοινωνίας) επάνω στην ήδη υπάρχουσα υποδομή του διαδικτύου [8]. Αυτή η πρόταση είναι βάσιμη αφού το HTTP αποτελεί το κυρίαρχο πρωτόκολλο στρώματος εφαρμογής στο διαδίκτυο με αποτέλεσμα όποια εφαρμογή το υιοθετήσει να έχει μεγάλη συμβατότητα με την ήδη υπάρχουσα δικτυακή υποδομή (software και hardware). Επιπλέον περιλαμβάνει αξιόπιστες και καλά δοκιμασμένες δυνατότητες κρυπτογράφησης (HTTPS).

Το HTTP χρησιμοποίει το μοντέλο επικοινωνίας request – response (αίτηση – απάντηση). Αυτό σημαίνει πως κάθε επικοινωνία αρχίζει με μία ερώτηση ενός κόμβου σε κάποιον άλλον με ένα αίτημα (request). Ο δεύτερος κόμβος αφού λάβει και επεξεργαστεί το αίτημα, αποκρίνεται με μία απάντηση (response). Κατά κύριο λόγο, το μοντέλο αυτό έχει ταυτιστεί με την αρχιτεκτονική client – server (πελάτη – εξυπηρετητή). Στην αρχιτεκτονική αυτή, οι κόμβοι διαχωρίζονται σε servers (εξυπηρετητές) και clients (πελάτες). Clients είναι αυτοί που στέλνουν requests στους servers, ενώ servers είναι αυτοί που αναλαμβάνουν να δέχονται τα requests και να αποκρίνονται με responses. Η αρχιτεκτονική αυτή, παρά τη δυσκολία που παρουσιάζει στην επίτευξη της αμφίδρομης επικοινωνίας, αποτελεί μια σταθερή, καθιερωμένη και ευρέως υποστηρίξιμη τεχνολογία. Οι παραπάνω παράγοντες συνεπώς, καθιστούν το HTTP μία απλή και εύκολη στην υλοποίηση λύση, για σχετικά μικρές εφαρμογές IoT.

**CoAP (Constrained Application Protocol)**

Η M2M δικτυακή κίνηση αποτελείται κυρίως από μικρές σε μέγεθος και με μεγάλη χρονική απόσταση μεταξύ τους μεταδόσεις δεδομένων [3]. Επιπλέον ένα μέρος της διαδρομής του δικτύου που ακολουθούν τα δεδομένα είναι μέσα σε δίκτυα χαμηλής ισχύος και με υψηλό ποσοστό απώλειας δεδομένων (Low power and Lossy Networks ή LLN) [8]. Δεδομένου των παραπάνω ένα κατάλληλο για την M2M επικοινωνία πρωτόκολλο πρέπει να μεγιστοποιεί την εξοικονόμηση ενέργειας, να είναι αποδοτικό όταν χρησιμοποιείται σε LLN's και να είναι σχετικά ελαφρύ, ώστε να εκτελείται σε μικρών δυνατοτήτων ενσωματωμένα συστήματα με περιορισμένη μνήμη και επεξεργαστική ισχύ. Το πρωτόκολλο HTTP μπορεί να έρθει σε αντιδιαστολή με τις απαιτήσεις της διαδικτυακής κίνησης του M2M καθώς διαχειρίζεται τη μετάδοση μεγάλων πακέτων δεδομένων με σταθερό ή μεταβλητό ρυθμό (λήψη αρχείων, VoIP) ή ακόμη παρουσιάζει διαστήματα αδράνειας όταν ο χρήστης περιηγείται στο διαδίκτυο.

Ο οργανισμός IETF συνέστησε την ομάδα εργασίας CoRE (Constrained RESTful Environments) προκειμένου να εργαστεί στην δημιουργία ενός πλαισίου λειτουργίας resource – oriented εφαρμογών σχεδιασμένων να λειτουργούν σε μικρά δίκτυα χαμηλής ενέργειας [9]. Η ομάδα εργασίας αυτή δημιούργησε το CoAP (Constrained Application Protocol), ένα πρωτόκολλο επιπέδου εφαρμογής σχεδιασμένο για χρήση σε περιορισμένων δυνατοτήτων δίκτυα και κόμβους.

Το CoAP μπορεί να αλληλοεπιδράσει εύκολα με το πρωτόκολλο HTTP, ώστε να υπάρχει δυνατότητα διασύνδεσης των μικρών IoT δικτύων με το διαδίκτυο, μέσω διαμεσολαβητών (proxies). Το βασικό μοντέλο αλληλεπίδρασης του CoAP είναι παρόμοιο με εκείνο του HTTP. Ένας client αποστέλλει ένα αίτημα με το οποίο αιτείται την εφαρμογή μιας ενέργειας (από τις GET, PUT, POST, DELETE) σε ένα resource στο server με τη μορφή μιας αναγνωριστικής σειράς χαρακτήρων (URI - Uniform Resource Identifiers) και ο server, αφού επεξεργαστεί το αίτημα, αποστέλλει μια απάντηση. Το CoAP σε αντίθεση με το HTTP, χειρίζεται αυτές τις ανταλλαγές μηνυμάτων ασύγχρονα χρησιμοποιώντας το UDP ως πρωτόκολλο επιπέδου μεταφοράς [10].

**MQTT (Message Queue Telemetry Transport)**

Το πρωτόκολλο αυτό, όπως και το CoAP, είναι σχεδιασμένο για κόμβους/συσκευές περιορισμένων δυνατοτήτων. Η αρχιτεκτονική είναι και εδώ client – server, ωστόσο, χρησιμοποιείται το μοντέλο επικοινωνίας publish – subscribe (έκδοσης – συνδρομής). Server αποτελεί ένας διακομιστής διαμεσολάβησης με τον οποίο συνδέονται όλοι οι κόμβοι στο δίκτυο ως clients. Τα έξυπνα αντικείμενα του δικτύου (things) αποτελούν τους publishers (εκδότες) που στέλνουν δεδομένα στο διαμεσολαβητή, ενώ οι ενδιαφερόμενοι clients για τα δεδομένα αυτά των things, αποτελούν τους subscribers (συνδρομητές).

Το μοντέλο αυτό επικοινωνίας χρήζει μεγάλης σημασίας σε IoT δίκτυα. Ιδιαίτερα αν θεωρήσουμε ότι η πλειονότητα των things έχουν περιορισμένες δυνατότητες επεξεργαστικής ισχύος ενώ οι ενδιαφερόμενοι κόμβοι για τα δεδομένα των things αποτελούν περισσότερο ικανές συσκευές, τότε είναι σημαντικό τα things να στέλνουν στους κόμβους ένα μόνο μήνυμα για κάθε ενημέρωση των δεδομένων του αισθητήρα τους, αντί να ειδοποιούν κάθε ενδιαφερόμενο ξεχωριστά, δυνατότητα την οποία παρέχει το πρωτόκολλο MQTT. Στην επόμενη υποενότητα που ακολουθεί, αναλύεται ο τρόπος λειτουργίας του.

### 2.1.3 MQTT

Το MQTT [11] όπως αναφέρθηκε, είναι ένα πρωτόκολλο δημοσίευσης – εγγραφής (publish - subscribe) μηνυμάτων για χρήση πάνω από TCP/IP, και προεπιλεγμένη θύρα τη 1883. Είναι ελαφρύ, απλό και εύχρηστο στην εφαρμογή του, χαρακτηριστικά που το καθιστούν ιδανικό για IoT εφαρμογές όπου το εύρος ζώνης του δικτύου είναι περιορισμένο και απαιτείται ένα “μικρό ίχνος κώδικα”. Ακόμη η χαμηλή κατανάλωση ενέργειας είναι άλλος ένας λόγος που συχνά επιλέγεται και για εφαρμογές κινητού.

Ένα θέμα για το οποίο γίνεται συζήτηση κινείται γύρω από τη σημασία του ακρωνυμίου MQTT. Η σύντομη απάντηση είναι ότι δεν δίνεται κάποιος ορισμός για το ακρωνύμιο πλέον και το πρωτόκολλο ονομάζεται απλά MQTT. Εκτενέστερα, ως ακρωνύμιο υπήρχε το MQ Telemetry Transport όπου το MQ αναφέρεται σε ένα προϊόν της IBM (International Business Machines), το MQ Series, το οποίο υποστήριζε το MQTT και από εκεί πήρε και το όνομα του το πρωτόκολλο. Εσφαλμένα στο διαδίκτυο παρατηρούμε να χαρακτηρίζεται και ως πρωτόκολλο

σειριακών μηνυμάτων (message queue protocol). Παρόλο που το πρωτόκολλο υποστηρίζει τη σειριοποίηση, δεν ακολουθεί την παραδοσιακή έννοια του σειριακού πρωτοκόλλου όπως σε άλλα παραδοσιακά συστήματα message queuing, καθώς δεν υπάρχουν ουρές (queues).

#### *2.1.3.1 Publish/Subscribe Μοντέλο*

Η φιλοσοφία του MQTT διαφέρει από την κλασική προσέγγιση πελάτη - εξυπηρετητή (client - server) όπου ένα πελάτης επικοινωνεί απευθείας με τον εξυπηρετητή και αντίστροφα. Το πρότυπο publish - subscribe (εκδότης - συνδρομητής) έρχεται ως εναλλακτική στο παραδοσιακό μοντέλο του client – server, καθώς εδώ ο πελάτης που στέλνει ένα μήνυμα δε γνωρίζει την ύπαρξη του παραλήπτη και η επικοινωνία βασίζεται στην ύπαρξη μιας ενδιάμεσης οντότητας.

Οι οντότητες που επικοινωνούν είναι εκδότες (publishers), είτε συνδρομητές (subscribers) ή και τα δύο. Η επικοινωνία γίνεται όταν οι publishers παράγουν μηνύματα τα οποία λαμβάνουν (καταναλώνουν) οι subscribers.

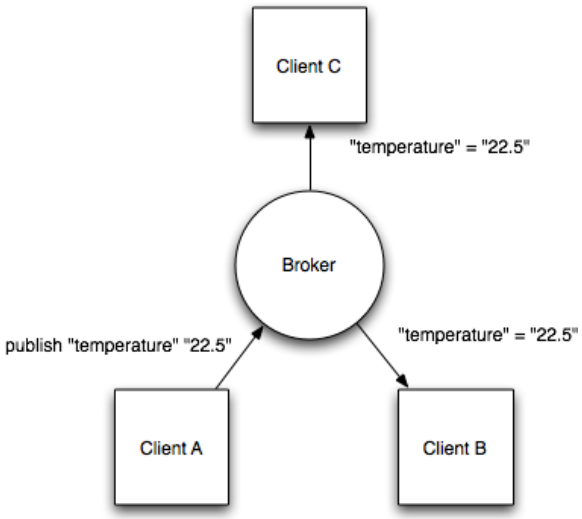
Κάθε μήνυμα ανήκει σε ένα θέμα (topic). Τα topics είναι συμβολοσειρές σε ιεραρχική δομή χωρισμένες με το χαρακτήρα slash (/). Ένας συνδρομητής εγγράφεται σε ένα ή περισσότερα θέματα που τον ενδιαφέρουν και λαμβάνει μόνο μηνύματα αυτών των θεμάτων. Αντίστοιχα ο εκδότης που παράγει ένα μήνυμα, το κατηγοριοποιεί σε ένα θέμα, οπότε μόνο οι συνδρομητές αυτού του θέματος το λαμβάνουν.

Την υλοποίηση της επικοινωνίας αναλαμβάνει μια ενδιάμεση οντότητα που ονομάζεται broker και είναι γνωστή τόσο στους εκδότες όσο και στους συνδρομητές. Λαμβάνει τα μηνύματα από τους εκδότες και τα προωθεί στους ενδιαφερόμενους συνδρομητές.

Ουσιαστικά με αυτό το μοντέλο πετυχαίνουμε 3 είδη αποσύνδεσης.

- Χωρική αποσύνδεση. Ο publisher και ο subscriber δε γνωρίζουν ο ένας την ύπαρξη του άλλου.
- Χρονική αποσύνδεση. Ο publisher και ο subscriber δε χρειάζεται να τρέχουν την ίδια χρονική στιγμή.
- Αποσύνδεση συγχρονισμού. Όλα τα μέρη (publisher, subscriber, broker) δε διακόπτουν την λειτουργία τους κατά την αποστολή ή παραλαβή μηνυμάτων.

Ο τρόπος που αποστέλλονται και φιλτράρονται τα μηνύματα είναι ο λόγος που επιτρέπει να υπάρχει αυτή η αποσύνδεση και προσδίδει στο MQTT ευελιξία στην ιεράρχηση μηνυμάτων και καλύτερη οργάνωση. Για την κατανόηση αυτού δίνεται ένα σχετικό παράδειγμα.




**Εικόνα 2-1: Publish/Subcribe Μοντέλο** [12]

Έστω ότι ο Client A στην εικόνα 2.1 βρίσκεται στο σαλόνι και θέλει να ενημερώσει την τιμή της θερμοκρασίας. Ένα πιθανό σενάριο είναι να στείλει την τιμή στο topic Home/livingroom/temperature, και όποιος θέλει να ενημερωθεί για αυτή τη θερμοκρασία πρέπει να κάνει subscribe στο αντίστοιχο topic.

Αυτή η δομή δίνει τη δυνατότητα στο συνδρομητή να εγγραφεί σε πολλά θέματα ταυτόχρονα με τη χρήση χαρακτήρων wildcards (#, +). Ο χαρακτήρας '#' τοποθετείται αυστηρά στο τέλος του topic που ζητείται και ταιριάζει όλα τα εναπομείναντα επίπεδα, επιτρέποντας μας να κάνουμε subscribe σε όλο το υποδέντρο. Πιο συγκεκριμένα με subscribe στο topic Home/livingroom/# θα λαμβάνονται τα δεδομένα τόσο του αισθητήρα θερμοκρασίας, αλλά και όποιας άλλης συσκευής-αισθητήρα στέλνει δεδομένα κάτω από το Home/livingroom/, π.χ Home/livingroom/humidity, Home/livingroom/light_switch κλπ.

Ο άλλος σημαντικός χαρακτήρας wildcard είναι ο χαρακτήρας '+'. Σε αντίθεση με τον '#', ο '+' αντικαθιστά ένα μόνο επίπεδο. Στην θέση του '+' θα αντικατασταθεί οτιδήποτε είναι γνωστό στο Broker ώστε να συμπληρωθεί κάποιο topic. Για παράδειγμα το Home/+/light_switch θα ταιριάξει και με το Home/bedroom/light_switch και με το Home/livingroom/light_switch,

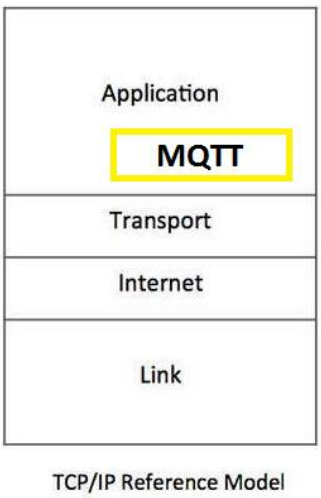
δημιουργώντας έτσι εδώ ένα σενάριο για το φωτισμό της οικίας που επηρεάζει και το υπνοδωμάτιο και το σαλόνι.

Ένα μεγάλο πλεονέκτημα του Pub/Sub μοντέλου έναντι του παραδοσιακού client-server είναι ότι οι διαδικασίες στο broker είναι υψηλά παραλληλοποιήσιμες και έτσι επιτρέπει στις εφαρμογές μας να κλιμακώνουν αρκετά. Επιπλέον η πολυεπίπεδη δομή των θεμάτων με σωστή σχεδίαση επιτρέπει και την ομαλή επέκταση του συστήματος όταν αυτό χρειαστεί. Ένα μειονέκτημα ίσως αποτελεί το ό,τι τόσο ο publisher όσο και ο subscriber πρέπει να είναι γνώστες της δομής των topics.

### *2.1.3.2 Σύνδεση client και broker*

**Clients**

Ένας MQTT client μπορεί να είναι οποιοδήποτε είδος συσκευής (από έναν απλό Micro controller έως και ένα σύγχρονο και πανίσχυρο μηχάνημα server) η οποία φέρει μια βιβλιοθήκη υλοποίησης του MQTT και μπορεί να συνδέεται μέσω οποιουδήποτε δικτύου με τον broker. Στην περίπτωση ενός περιβάλλοντος internet (TCP/IP) η βιβλιοθήκη MQTT υλοποιείται στο επίπεδο εφαρμογής όπως και άλλα παρόμοια πρωτόκολλα πχ HTTP (Εικόνα 2.2).




**Εικόνα 2-2: Η θέση του πρωτοκόλλου MQTT στην ιεραρχία επιπέδων του TCP/IP**

**Broker**

Η καρδιά οποιουδήποτε pub/sub μοντέλου είναι ο broker ο οποίος λαμβάνει όλα τα μηνύματα, τα φιλτράρει και αποφασίζει σε ποιους συνδρομητές θα τα προωθήσει. Ταυτοποιεί επίσης όλους τους clients μέσω μεθόδων αυθεντικοποίησης (authentication method) και ελέγχει για εξουσιοδοτημένους συνδρομητές αν έχει προστεθεί τέτοια υλοποίηση. Επιπρόσθετα κρατά όλα τα δεδομένα που αφορούν τους clients που έχουν κάνει ''επίμονη'' σύνδεση (persisted clients). Πιο κάτω παρατίθενται περισσότερες πληροφορίες γι' αυτό.

Ο broker όντας ο server του συστήματος, θα πρέπει να επιτρέπει εύκολη κλιμάκωση (scalability) ώστε να μπορεί να ανταπεξέλθει στο φόρτο εργασίας. Έτσι λοιπόν ανάλογα με την υλοποίηση μπορεί να διαχειριστεί ταυτόχρονα χιλιάδες συνδέσεις.

**Εγκατάσταση Σύνδεσης**

Η MQTT σύνδεση εγκαθιδρύεται πάντα μεταξύ ενός client και του broker και αρχικοποιείται με τον client να στέλνει ένα CONNECT μήνυμα στον broker. Εφόσον είναι εφικτή, ακολουθεί η απάντηση του broker με ένα CONNACK (Connection Acknowledgment) μήνυμα και την κατάσταση σύνδεσης. Ο broker θα διατηρήσει τη σύνδεση μέχρι να λάβει κάποια εντολή αποσύνδεσης από τον client. Ένα CONNECT περιλαμβάνει τις εξής πληροφορίες:

- clientId – Ένα μοναδικό αναγνωριστικό για κάθε MQTT client. Αν δεν οριστεί από το client ο broker θα του δώσει έναν τυχαίο αριθμό.
- Clean Session – Αυτή η μεταβλητή καθορίζει αν η σύνδεση του client θα είναι μόνιμη (persistent session). Μία τέτοιου τύπου σύνδεση (Clean Session = false) σημαίνει ότι θα αποθηκεύονται στον broker όλα τα subscriptions και τα χαμένα μηνύματα (με QoS>0) που αφορούσαν αυτόν τον client και για κάποιο λόγο δεν παραδόθηκαν (πιθανή αποσύνδεση).
- Username/Password – Επιτρέπουν την ταυτοποίηση του client σε περίπτωση που κάτι τέτοιο απαιτείται.
- Will Message – Αυτή η λειτουργία επιτρέπει την ενημέρωση των clients έπειτα από μια ξαφνική αποσύνδεση ενός client από το σύστημα. Όταν αντιληφθεί ο broker ότι ο εκδότης client έχει πάψει να λειτουργεί προωθεί ένα μήνυμα στους ενδιαφερόμενους clients.

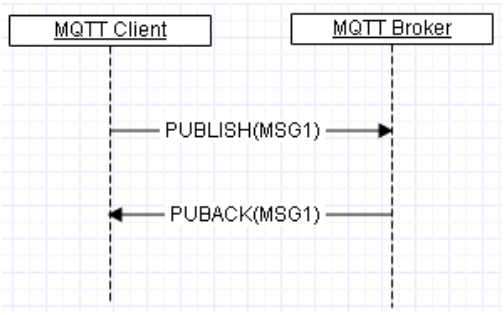
- KeepAlive – Πρόκειται για το χρονικό περιθώριο στο οποίο δεσμεύεται ο client να στέλνει ένα PING Request και ο broker να στέλνει ένα PING Response, ώστε να επιβεβαιώνεται και από τις δύο πλευρές ότι η σύνδεση είναι ενεργή.

#### *2.1.3.3 Quality of Services (QoS)*

Η ποιότητα παρεχόμενης υπηρεσίας (Quality of Services) χρησιμοποιείται στα συστήματα επικοινωνίας για να δηλώσει την απόδοση ενός συστήματος όπως την αντιλαμβάνονται οι χρήστες του. Στο MQTT, το QoS είναι μια συμφωνία μεταξύ του αποστολέα και του παραλήπτη για το πόσο εγγυημένη είναι η παράδοση ενός μηνύματος στον προορισμό του.

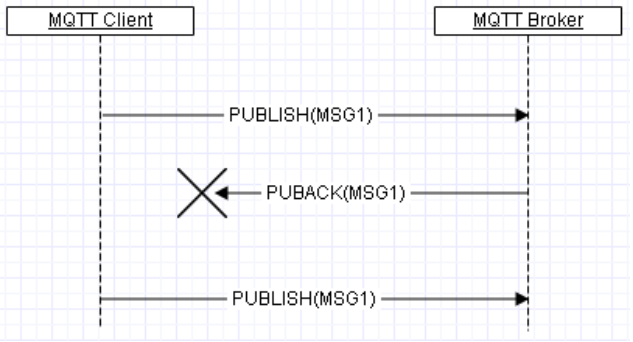
Η δημοσίευση ενός μηνύματος από έναν client-εκδότη, όπως αναλύθηκε, συνεπάγεται ότι το μήνυμα αυτό θα φθάσει στους client-συνδρομητές σε δύο βήματα: (α) από τον client-εκδότη στον broker και (β) από τον broker στον client-συνδρομητή. Για το βήμα (α), το QoS καθορίζεται από τον publisher (εντολή publish) και ορίζεται για κάθε μήνυμα. Για το (β), το QoS ορίζεται με την εγγραφή του συνδρομητή στο συγκεκριμένο θέμα (εντολή subscribe). Συνεπώς, το QoS μπορεί να υποβιβαστεί για κάποιο παραλήπτη ο οποίος έκανε subscribe με μικρότερο QoS. Στο MQTT υπάρχουν τρία QoS επίπεδα:

QoS(0): Το ελάχιστο επίπεδο QoS, το μήνυμα αποστέλλεται μια φορά και δε λαμβάνεται επιβεβαίωση παράδοσης, ούτε αποθηκεύεται. Συνήθως είναι όσο αξιόπιστο είναι και το TCP πρωτόκολλο πάνω από το οποίο γίνεται η αποστολή. Αυτό το επίπεδο επιλέγεται όταν υπάρχει σταθερή σύνδεση broker και client (συνήθως ενσύρματη).



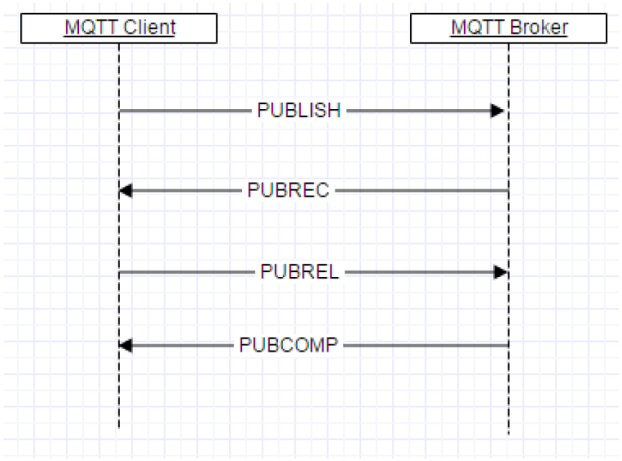

**Εικόνα 2-3: Δημοσίευση μηνύματος με QoS = 0**

QoS(1): Η επιλογή 1 εγγυάται ότι το μήνυμα θα παραδοθεί τουλάχιστον μια φορά στον παραλήπτη. Για κάθε μήνυμα που λαμβάνει ο broker στέλνει μήνυμα επιβεβαίωσης (puback) στον αποστολέα που σημαίνει ότι το μήνυμα έφθασε στον προορισμό του. Αν περάσει συγκεκριμένο χρονικό διάστημα και δεν λάβει το PUBACK μήνυμα, ο αποστολέας προσπαθεί να ξαναστείλει το ίδιο μήνυμα. Με αυτόν το τρόπο υπάρχει μεγάλη πιθανότητα το μήνυμα να φτάσει περισσότερες φορές στον παραλήπτη. Αυτό το επίπεδο επιλέγεται όταν σκοπός είναι να φτάσει σίγουρα η πληροφορία και εναπόκειται στο χρήστη αν θα χειριστεί την επιπρόσθετη πληροφορία (τα αντίγραφα).




**Εικόνα 2-4: Δημοσίευση μηνύματος με QoS = 1**

QoS (2): Το επίπεδο αυτό εγγυάται ότι το μήνυμα θα παραδοθεί ακριβώς μια φορά στον παραλήπτη. Η εγγύηση προέρχεται από δύο ροές μηνυμάτων (διπλή επικοινωνία) μεταξύ αποστολέα και παραλήπτη με συνολικά τέσσερα βήματα (4-step handshake) όπως φαίνεται και στην εικόνα 2.5.




**Εικόνα 2-5: Δημοσίευση μηνύματος με QoS = 2**

Ο broker όταν λάβει μήνυμα PUBLISH, το κρατάει μέχρι να στείλει το πακέτο PUBCOMP. Αν ο client στείλει δεύτερη φορά ένα μήνυμα, γιατί καθυστέρησε η απάντηση, ο broker θα προωθήσει μόνο το ένα. Αυτό το επίπεδο επιλέγεται όταν είναι πολύ σημαντικό για την εφαρμογή να λαμβάνονται όλα τα μηνύματα ακριβώς μια φορά. Είναι το πιο αργό και απαιτητικό σε πόρους, αφού απαιτεί περισσότερη επικοινωνία αλλά και χρήση μνήμης μέχρι την ολοκλήρωση της ενέργειας, αλλά η επιβάρυνση αυτή από το διπλό έλεγχο δεν αποτελεί τόσο μειονέκτημα.

#### *2.1.3.4 Επίμονη σύνδεση και σειριοποίηση μηνυμάτων*

Όταν ένας client συνδέεται στο broker πρέπει να κάνει subscribe σε όλα τα topics που τον ενδιαφέρουν. Κατά την επανασύνδεση αυτή, η διαδικασία πρέπει να επαναληφθεί, αν η σύνδεση δεν είναι ”επίμονη”. Για να γίνει μία σύνδεση επίμονη (persistent session) πρέπει κατά τη σύνδεση να τεθεί η μεταβλητή cleanSession ως ψευδής. Αν υπάρχει ήδη αυτός ο client αποθηκευμένος, πρόκειται για επανασύνδεση. Σε αυτήν την περίπτωση θα του αποσταλεί ό,τι πληροφορία έχει αποθηκεύσει ο broker για αυτόν. Στην πληροφορία συμπεριλαμβάνονται η ύπαρξη της σύνδεσης, προηγούμενα subscriptions και όλα τα μηνύματα με QoS 1 ή 2 τα οποία δεν έχουν επιβεβαιωθεί ότι παρελήφθησαν από τον client. Για να διαγραφεί μια επίμονη σύνδεση από το broker πρέπει ο ίδιος client (clientID) να κάνει σύνδεση με cleanSession αληθές.

Από τα πιο πάνω συμπεραίνει κανείς ότι το MQTT μπορεί να χρησιμοποιηθεί σαν κλασσικό πρωτόκολλο σειριακών μηνυμάτων, αφού ο client έχει την επιλογή να αποθηκεύονται όλα τα μηνύματα όταν δεν είναι συνδεδεμένος και να τα παραλαμβάνει όταν συνδεθεί και πάλι. Αυτό όμως που κάνει το MQTT να διαφέρει παρουσιάζεται στην παρακάτω σύγκριση.

Στο κλασσικό πρωτόκολλο σειριακών μηνυμάτων τα μηνύματα σειριοποιούνται μέχρι κάποιος καταναλωτής-client να τα παραλάβει. Αν δεν υπάρξει παραλαβή θα παραμείνουν ”κολλημένα” στη σειρά περιμένοντας να καταναλωθούν. Αντίθετα στο πρωτόκολλο MQTT δεν υπάρχει ο περιορισμός αυτός. Στην περίπτωση που δεν έχει γίνει subscribe σε αυτό το topic από κάποιον client, το μήνυμα δε θα σταλθεί σε κανέναν αφού δεν υπάρχει ενδιαφερόμενος παραλήπτης. Επιπλέον, μεγάλη διαφορά είναι ότι στο κλασσικό πρωτόκολλο σειριακών μηνυμάτων κάθε μήνυμα έχει μονάχα

έναν παραλήπτη ενώ όπως είδαμε στο MQTT παραλήπτες μπορεί να είναι όσοι έχουν κάνει subscribe στο αντίστοιχο topic.

#### *2.1.3.5 Ασφαλής σύνδεση με το MQTT*

**Πιστοποίηση**

Το MQTT πρωτόκολλο παρέχει την επιλογή για πιστοποίηση κατά την σύνδεση στον broker (πεδία username,password). Η αποστολή usename και password στον broker, προϋποθέτει την κατάλληλη ρύθμιση του τελευταίου, με την υλοποίηση αρχείου με τους εγκεκριμένους χρήστες, ώστε να γίνει η επαλήθευση των στοιχείων τους κατά τη σύνδεση. Θα πρέπει να επισημανθεί ωστόσο ότι τα στοιχεία στέλνονται σε απλό κείμενο. Για να αποτραπεί οποιαδήποτε πιθανότητα κλοπής κατά τη μεταφορά τους, τα δεδομένα μπορούν να κρυπτογραφηθούν.

Άλλη μια μέθοδος είναι η χρήση πιστοποιητικού όπως το X.509 το οποίο παρουσιάζεται στο broker κατά την TLS χειραψία. Αρκετοί brokers επιτρέπουν να χρησιμοποιηθεί η πληροφορία από το πιστοποιητικό σε επίπεδο εφαρμογής, αφού όμως προηγηθεί επιτυχής χειραψία.

**Κρυπτογράφηση**

Η πιστοποίηση είναι πρακτική ασφαλείας σε επίπεδο εφαρμογής. Σε επίπεδο μεταφοράς δεδομένων στο MQTT υπάρχει η δυνατότητα εφαρμογής πρωτοκόλλων κρυπτογραφίας TLS (Transport Layer Security) και SSL (Secure Sockets Layer), με χρήση της θύρας 8883. Τα πρωτόκολλα TLS/SSL χρησιμοποιούν μηχανισμούς χειραψίας ώστε να ρυθμίσουν κατάλληλες παραμέτρους για να δημιουργήσουν μια ασφαλή σύνδεση ανάμεσα σε client και server. Σε αυτή τη σύνδεση πλέον τα δεδομένα μεταφέρονται κρυπτογραφημένα και με την παρούσα τεχνολογία θεωρείται πολύ δύσκολο να διαβαστούν.

Συστήνεται ειδικά όταν γίνεται χρήση username και password. Στην περίπτωση που η συσκευή δεν μπορεί να στηρίξει TLS σύνδεση, εναλλακτική μέθοδος είναι να κρυπτογραφηθούν τόσο τα μηνύματα (το πεδίο payload στην εντολή publish) όσο και ο κωδικός (στην εντολή connect) σε επίπεδο εφαρμογής.

## 2.2 Μικροελεγκτές (Single-board microcontrollers)

### 2.2.1 Δομή Μικροελεγκτή

Ο μικροελεγκτής είναι ένα πλήρες υπολογιστικό σύστημα βελτιστοποιημένο για τον έλεγχο υλικού (hardware) και ενσωματώνει όλες τις διακριτές μονάδες (μικροεπεξεργαστή, μνήμη και περιφερειακές μονάδες εισόδου/εξόδου I/O ports) μέσα στο σώμα ενός μόνο ολοκληρωμένου κυκλώματος. Η ύπαρξη των παραπάνω σε μια και μόνο ψηφίδα πυριτίου σημαίνει ότι η ταχύτητα ενισχύεται, καθώς οι περιφερειακές μονάδες εισόδου/εξόδου απαιτούν λιγότερο χρόνο να διαβάσουν ή να γράψουν από τις εξωτερικές συσκευές. Επιπλέον ο επεξεργαστής και η μνήμη ανταλλάσσουν δεδομένα γρηγορότερα.

Ένας μικροελεγκτής είναι βελτιστοποιημένος για να συντονίζει την ροή των δεδομένων μεταξύ των μονάδων μνήμης και των περιφερικών συσκευών εκτός του περιβάλλοντος του. Οι συνδέσεις ενός μικροεπεξεργαστή περιλαμβάνουν τη διευθυνσιοδότηση και τους διαύλους δεδομένων, που του επιτρέπουν να επιλέξει ένα από τα περιφερειακά του και να στείλει ή να ανακτήσει δεδομένα από αυτά. Επειδή ο επεξεργαστής του μικροελεγκτή και τα περιφερειακά του είναι ενσωματωμένα στην ίδια ψηφίδα πυριτίου, οι μονάδες που περιέχει είναι αυτόνομες και σπάνια έχουν δομές διαύλων που εκτείνονται έξω από αυτούς.

Ένας μικροελεγκτής, αποτελείται κυρίως από:

- τη μονάδα επεξεργασίας (CPU) ή μικροεπεξεργαστή
- τη μνήμη
- τις θύρες εισόδου/εξόδου (I/O ports)

Πλέον, οι μικροελεγκτές διαθέτουν ενσωματωμένα και αρκετά περιφερειακά που τους δίνουν αυξημένες δυνατότητες. Για παράδειγμα, ο μετατροπέας από αναλογικό σε ψηφιακό (ADC), η δυνατότητα διαμόρφωσης διάρκειας εύρους παλμών (Pulse Width Modulation, PWM) εξόδου αλλά και οι τρόποι επικοινωνίας με διάφορα ηλεκτρονικά συστήματα καθορίζουν την τελική εκλογή του μικροελεγκτή. Τέλος, η γλώσσα προγραμματισμού και τα εργαλεία ανάπτυξης γίνονται πλέον και με γλώσσες υψηλού επιπέδου προσφέροντας αυξημένες δυνατότητες.

### 2.2.2 Οι επεξεργαστές ARM

Η πρώτη ερμηνεία των ακρωνύμιου ARM που δόθηκε ήταν Acorn RISC Machine, ενώ η μεταγενέστερη αφορά τον όρο Προχωρημένη Μηχανή RISC (Advanced RISC Machine). Οι επεξεργαστές ARM ακολουθούν αρχιτεκτονική συνόλου εντολών RISC(Reduced Instruction Set Computing) και αποτελούν τη δημοφιλέστερη αρχιτεκτονική συστημάτων 32-bit. Αυτό σημαίνει πως οι επεξεργαστές ARM χρειάζονται πολύ λιγότερα transistors από την τυπική CISC (Complex Instruction Set Computing) που χρησιμοποιούν οι επεξεργαστές των ηλεκτρονικών υπολογιστών. Αυτό μειώνει το κόστος, την θερμοκρασία του επεξεργαστή και την κατανάλωση, με αποτέλεσμα η χρήση τους να είναι κατάλληλη για εφαρμογές χαμηλής ισχύος. Οι μικροεπεξεργαστές ARM έχουν υπερισχύσει στις αγορές των κινητών και των ενσωματωμένων συστημάτων [13], σαν μικροί και σχετικά χαμηλού κόστους μικροεπεξεργαστές [14].

#### *2.2.2.1 Αρχιτεκτονική RISC*

Reduced Instruction Set Computer (RISC) είναι μία στρατηγική σχεδίαση της Κεντρική Μονάδας Επεξεργασίας που βασίζεται στο γεγονός ότι ένα απλοποιημένο σύνολο εντολών θα μεγιστοποιήσει την απόδοση ενός συστήματος βασισμένο σε έναν μικροεπεξεργαστή ο οποίος εκτελεί συγκεκριμένες εντολές σε λιγότερους κύκλους ανά εντολή.

Τα στοιχεία που διατηρήθηκαν στον ARM είναι τα εξής

- Αρχιτεκτονική φόρτωσης/αποθήκευσης (load/store). Οι εντολές οι οποίες επεξεργάζονται τα δεδομένα λειτουργούν κατευθείαν στους καταχωρητές του επεξεργαστή και είναι ξεχωριστές από τις εντολές προσπέλασης της μνήμης.
- Περιλαμβάνει καταχωρητές με σταθερό μέγεθος εντολών για εύκολη αποκωδικοποίηση και διοχέτευση (pipeline).
- Ο χρόνος εκτέλεσης των εντολών γίνεται σε ένα κύκλο ρολογιού.

### 2.2.3 Οι μικροελεγκτές AVR

Οι AVR [15] είναι μικροελεγκτές με επεξεργαστή RISC ο οποίος έχει σχεδιαστεί με βάση την αρχιτεκτονική Harvard. Στην αρχιτεκτονική αυτή υπάρχει ξεχωριστός δίαυλος για τη μεταφορά των δεδομένων (data bus) και ξεχωριστός για την μεταφορά των εντολών (instruction bus). Η ύπαρξη δύο διαφορετικών μνημών, αυτή των

δεδομένων (data memory) και του προγράμματος (program memory), καθιστά την αρχιτεκτονική Harvard ιδιαίτερα αποδοτική, καθώς οι εντολές εκτελούνται παράλληλα με την εγγραφή ή την προσπέλαση της μνήμης. Με αυτόν τον τρόπο, εκτελούνται περισσότερες εντολές σε έναν κύκλο ρολογιού.

Το σημαντικό σε όλους τους τύπους μικροελεγκτών AVR είναι ότι ο πυρήνας της αρχιτεκτονικής τους είναι ο ίδιος και έχουν το ίδιο σύνολο εντολών. Διαφέρουν μόνο στα περιφερειακά και στο μέγεθος της μνήμης που παρέχουν. Συνεπώς, το λογισμικό είναι μεταφέρσιμο μεταξύ των διάφορων τύπων. Αν ένας μικροελεγκτής διαθέτει μεγαλύτερη πληθώρα περιφερειακών, απαιτείται η εισαγωγή του αντίστοιχου κώδικα για τη λειτουργίας τους.

## 2.3 Αισθητήρες και μέτρηση ηλεκτρικών μεγεθών

Ο αισθητήρας είναι μία διάταξη που χρησιμοποιείται για την μέτρηση ενός φυσικού μεγέθους. Η βασική λειτουργία του είναι η ανίχνευση ενός σήματος ή μιας διέγερσης και η παραγωγή μιας μετρήσιμης εξόδου. Διαθέτουν κάποια κατάλληλη ιδιότητα, η οποία μεταβάλλεται ως συνάρτηση του μετρούμενου φυσικού μεγέθους. Επιτρέπει να αντιληφθεί κανείς και να ποσοτικοποιήσει διάφορα φυσικά μεγέθη και τις μεταβολές αυτών. Συνήθως μετατρέπει το μετρούμενο φυσικό μέγεθος σε ηλεκτρικό σήμα (τάση, ένταση, συχνότητα). Το σήμα εξόδου μπορεί να είναι τάση ή ρεύμα, ενώ το πλάτος, η συχνότητα, η φάση ή ο ψηφιακός κώδικας είναι χαρακτηριστικά που συμπληρώνουν το χαρακτηρισμό του. Επομένως ένας αισθητήρας έχει ιδιότητες εισόδου και ηλεκτρικές ιδιότητες εξόδου.

Μερικά παραδείγματα φυσικών μεγεθών που μετρούν οι αισθητήρες είναι η θερμοκρασία, η θέση και η μετατόπιση ενός αντικειμένου, η στάθμη υγρών, η ταχύτητα και η επιτάχυνση ενός κινούμενου αντικειμένου, η δύναμη, η ροή ρευστού, η τάση, το ρεύμα, η υγρασία, η ακτινοβολία κα.

### 2.3.1 Μέτρηση Ρεύματος

Καθοριστικό κομμάτι στη διεκπεραίωση της διαδικασίας μέτρησης του ρεύματος, αποτελεί η χρήση κάποιου αισθητήρα ή μορφομετατροπέα (transducer) ρεύματος. Η διάκριση αυτή γίνεται γιατί ο αισθητήρας μετατρέπει οποιαδήποτε μορφή ενέργειας σε ηλεκτρική, ενώ ο transducer μία μορφή ενέργειας σε οποιαδήποτε άλλη. Το παραγόμενο σήμα μπορεί να είναι αναλογικό ρεύμα, αναλογική τάση ή ακόμα και

ψηφιακό σήμα. Η είσοδος μπορεί να είναι είτε εναλλασσόμενη, είτε συνεχές ρεύμα και ανάλογα την εφαρμογή επιλέγουμε τον κατάλληλο αισθητήρα [16].

Δύο από τις πιο συνηθισμένες τεχνολογίες μέτρησης ρεύματος είναι οι:

- Μετασχηματιστές ρεύματος (transformers)
- Hall Effect αισθητήρες

### *2.3.1.1 Μετασχηματιστές ρεύματος*

Ο πιο διαδεδομένος τρόπος μέτρησης ρεύματος σε συστήματα ισχύος είναι μέσω μετασχηματιστή ρεύματος (Current Trasformer – CT). Ο σκοπός ενός μετασχηματιστή ρεύματος, είναι να μετατρέψει το ρεύμα του πρωτεύοντος ενός συστήματος ισχύος σε ένα σήμα που μπορεί να αξιοποιηθεί από ηλεκτρομηχανικές και ηλεκτρονικές συσκευές όπως προστατευτικοί ηλεκτρονόμοι, transducers κ.ά. Λόγω του ότι υπάρχει ένα μεγάλο εύρος συσκευών που μπορεί να συνδεθεί, τα χαρακτηριστικά του εκάστοτε μετασχηματιστή παίζουν καθοριστικό ρόλο στην επιλογή του κατάλληλου για την εκάστοτε εφαρμογή [17].

Ο μετασχηματιστής συνδέεται σε σειρά με την συσκευή που παράγει το ρεύμα προς μέτρηση και μετατρέπει το ρεύμα του πρωτεύοντος στο δευτερεύον τύλιγμα, σύμφωνα με τη σχέση :

$$\mathrm{I_P} = n \cdot I_S \quad (1)$$

όπου

- $\mathrm{I_p}$ το ρεύμα του πρωτεύοντος
- $\mathrm{I_s}$ το ρεύμα του δευτερεύοντος
- n ο λόγος των τυλιγμάτων μεταξύ πρωτεύοντος και δευτερεύοντος. Δηλαδή:

$$n = \frac{N_s}{N_p}$$

Με την τοποθέτηση μιας αντίστασης φορτίου RB (burden resistance) στο δευτερεύον τύλιγμα, της οποίας η επιλογή καθορίζει το πλάτος της τάσης εξόδου που θα παραχθεί από τον μετασχηματιστή, προκύπτει από το νόμο του Ohm ότι :

$$V_O = R_B \cdot I_S \quad (2)$$

Συνδυάζοντας τη σχέση (2) με την (1) που συνδέει το ρεύμα πρωτεύοντος και δευτερεύοντος, εξάγουμε :

$$V_O = R_B \cdot \frac{I_P}{n}$$

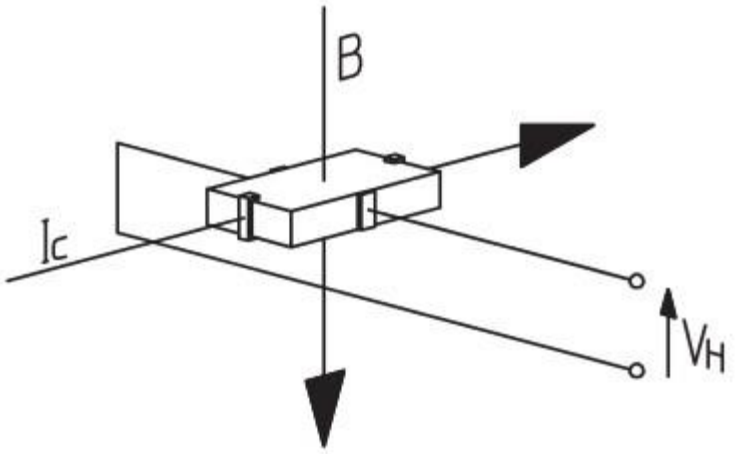
Από τον παραπάνω τύπο μπορούμε να καθορίσουμε την τιμή της τάσης εξόδου, ανάλογα με την εφαρμογή που χρειαζόμαστε.

**Hall Effect transducers**

Οι Hall Effect transducers [18] βασίζονται στο φαινόμενο Hall το οποίο προκαλείται από τις δυνάμεις Lorentz που επιδρούν σε κινούμενα φορτία μέσα σε ένα μαγνητικό πεδίο[19]. Η δύναμη είναι ίση με:

$$F_L = q \cdot (VXB)$$

Η αναπαράσταση του φαινομένου Hall φαίνεται στην εικόνα 2.6.




**Εικόνα 2-6: Το φαινόμενο Hall** [18]

Ένα λεπτό φύλλο επαγωγικού υλικού διαρρέεται από ρεύμα Ic. Τα κινούμενα ηλεκτρικά φορτία του ρεύματος επηρεάζονται από την εξωτερική μαγνητική ροή B, που παράγει μία δύναμη Lorentz FL, κάθετη στην κατεύθυνση του ρεύματος. Το αποτέλεσμα της εκτροπής του ρεύματος, είναι να δημιουργηθούν περισσότερα ηλεκτρικά φορτία που συγκεντρώνονται στην άκρη του φύλλου, δημιουργώντας μία διαφορά δυναμικού, την τάση Hall, $V_H$. Για την παραπάνω διάταξη, εξάγουμε τη σχέση :

$$V_H = \frac{K}{d} \cdot I_C \cdot B \cdot V_{OH}$$

όπου

- K - η σταθερά Hall του επαγόμενου υλικού
- d - το πάχος του φύλλου
- Ic - το ρεύμα που διαπερνά το φύλλο
- B - η ένταση του μαγνητικού πεδίου
- $V_{OH}$ - η Hall τάση με την απουσία εξωτερικού μαγνητικού πεδίου

#### *2.3.1.2 Ενεργός Τιμή (RMS Value) μιας Κυματομορφής AC*

Η ενεργός τιμή ενός σήματος ή αλλιώς RMS τιμή από το Root Mean Square, είναι μία ένδειξη του πλάτους ενός σήματος. Βρίσκει εφαρμογή κυρίως σε εναλλασσόμενα μεγέθη, δηλαδή ποσότητες που άλλοτε είναι αρνητικές και άλλοτε θετικές.

Οι αισθητήρες ρεύματος μπορούν να μετρήσουν συνεχή ρεύματα (DC) και εναλλασσόμενα ρεύματα (AC). Στο συνεχές σήμα δεν έχει νόημα να μετρούνται ενεργές τιμές και η ισχύς υπολογίζεται από τις μέσες τιμές των μεγεθών. Αντίθετα, στο εναλλασσόμενο σήμα AC, σε μια μέτρηση η έξοδος του αισθητήρα δίνει αρνητικές και θετικές τιμές και η κυματομορφή εξόδου είναι ανάλογη της εισόδου, του AC που μετριέται. Σε αυτήν την περίπτωση για τη λήψη σωστών ενδείξεων πρέπει να υπολογισθούν οι RMS τιμές της κυματομορφής εξόδου [20].

Η ενεργός τιμή της εναλλασσόμενης τάσης ή RMS τιμή (Root-Mean-Squared) ορίζεται ως η τιμή συνεχούς τάσης που παράγει την ίδια μέση ισχύ με την εναλλασσόμενη.

Γενικά, για κάθε περιοδική κυματομορφή V(t) η ενεργός τιμή $V_{RMS}$ , εξαρτάται από την περίοδο T μέσω της σχέσης:

$$V_{rms} = \sqrt{\frac{1}{T}\int_0^T V^2(t)dt}$$

Στην περίπτωση μιας ημιτονοειδούς κυματομορφής (εναλλασσόμενο ρεύμα AC), η ενεργός τιμή της εναλλασσόμενης τάσης δίνεται από τη σχέση:

$$V_{RMS} = V_{pk}\frac{1}{\sqrt{2}} = V_{pk} \times 0.7071$$

### 2.3.2 Μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to digital converters - ADC)

Στα συστήματα συλλογής και μετατροπής δεδομένων, είναι απαραίτητη η είσοδος σημάτων από μία ή περισσότερες πηγές και η μετατροπή τους σε ψηφιακά σήματα για ανάλυση ή μετάδοσή τους σε μία τερματική συσκευή ή την εμφάνιση τους σε μία ψηφιακή οθόνη. Τις περισσότερες φορές τα αναλογικά σήματα εισέρχονται στο σύστημα μέσω αισθητήρων οι οποίοι μετατρέπουν τα πραγματικά μεγέθη σε ηλεκτρικά σήματα και τα οποία με την σειρά τους ψηφιοποιούνται με τη βοήθεια ενός μετατροπέα.

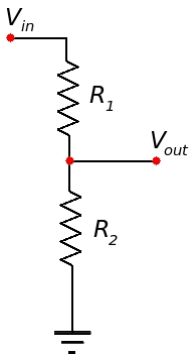
Οι μετατροπείς αναλογικού σήματος σε ψηφιακό (Analog-to-digital converters - ADC)[21] μεταφράζουν ένα αναλογικό σήμα ή ποσότητα, τα οποία είναι χαρακτηριστικά των φαινομένων στον πραγματικό κόσμο, στην ψηφιακή γλώσσα, που χρησιμοποιείται στην επεξεργασία των πληροφοριών, την πληροφορική, τη μετάδοση δεδομένων, καθώς και τα συστήματα ελέγχου. Η αντίθετη διαδικασία εκτελείται από μετατροπείς ψηφιακού σε αναλογικό (Digital-to-Analog Converters - DAC) που χρησιμοποιούνται στο μετασχηματισμό, μετάδοση και αποθήκευση των αποτελεσμάτων της ψηφιακής επεξεργασίας, πίσω στον πραγματικό κόσμο.

Το πιο σημαντικό στοιχείο των A/D μετατροπέων είναι το γεγονός ότι η έξοδος του σήματος είναι σε ψηφιακή μορφή, επομένως το σήμα εξέρχεται από τον μετατροπέα κβαντισμένο. Με άλλα λόγια, μία λέξη μεγέθους N-bit μπορεί να έχει μόνο $2^N$ πιθανές καταστάσεις. Επομένως, ένας N-bit μετατροπέας έχει μόνο $2^N$ πιθανές ψηφιακές εξόδους [22].

### 2.3.3 Διαιρέτης Τάσης

Η σύνδεση της εξόδου ενός αισθητήρα με την είσοδο του συστήματος συνήθως γίνεται με ένα διαιρέτη τάσης ο οποίος φροντίζει να προσαρμόσει την τάση εξόδου του αισθητήρα στα επιθυμητά όρια της τάσης εισόδου του μετατροπέα.

Ο διαιρέτης τάσης αποτελεί την πιο διαδεδομένη μέθοδο που χρησιμοποιείται σε εφαρμογές ψηφιακών μετρητικών για τη μέτρηση της τάσης DC και AC. Πρόκειται για μία απλή κυκλωματική διάταξη η οποία αποτελείται από δύο αντιστάσεις σε σειρά, στα άκρα των οποίων εφαρμόζεται η τάση εισόδου. Ως τάση εξόδου λαμβάνεται η διαφορά δυναμικού ανάμεσα στους ακροδέκτες της μιας εκ των δύο αντιστάσεων.




**Εικόνα 2-7: Κυκλωματικό διάγραμμα διαιρέτη τάσης διάγραμμα διαιρέτη τάσης**

Σύμφωνα με την θεωρία των ηλεκτρικών κυκλωμάτων η ένταση που διαρρέει το κύκλωμα των δύο αντιστάσεων είναι:

$$I = \frac{V_{in}}{R_1 + R_2}$$

Η τάση εξόδου θα είναι:

$$V_{out} = I \cdot R_2 \Rightarrow V_{out} = R_2 R_1 + R_2 \cdot V_{in}$$

Από την παραπάνω ανάλυση προκύπτει πως με κατάλληλη επιλογή των τιμών των ηλεκτρικών αντιστάσεων η τάση εξόδου μπορεί να είναι μέσα στο εύρος εισόδου των ολοκληρωμένων που χρησιμοποιούνται για την εκάστοτε μέτρηση των μεγεθών. Με αυτόν τον τρόπο προσαρμόζουμε την, υπό μέτρηση, τάση εισόδου σε επιτρεπτά όρια για τον μετατροπέα.

# 3 Έξυπνο Σπίτι

Η δικτυωμένη φύση των αναπτυσσόμενων ηλεκτρονικών συσκευών σε μία οικία σε συνδυασμό με τις νέες τεχνολογίες του διαδικτύου των αντικειμένων, έχουν δώσει το έναυσμα για τη δημιουργία πολλών ερευνητικών έργων που εντάσσονται στον τομέα του έξυπνου σπιτιού (smart home) .

Με τον όρο «Έξυπνο σπίτι» μπορεί να περιγραφεί οποιοδήποτε προσωπικό ή εργασιακό περιβάλλον που περικλείει ένα σύνολο τεχνολογικών εφαρμογών με κύριο χαρακτηριστικό την αυτοματοποίηση και τον έλεγχο των επιμέρους τμημάτων του. Ο βαθμός αυτοματοποίησης, καθώς επίσης και ο τρόπος ελέγχου ποικίλουν, αφού εξαρτώνται από πολλές παραμέτρους όπως το κόστος, το είδος των συσκευών που πρόκειται να ελεγχθούν, τον τύπο του κτηρίου στο οποίο θα εγκατασταθεί η τεχνολογία, τις προσωπικές ανάγκες του χρήστη.

Τα σύγχρονα συστήματα που εφαρμόζονται στις «Έξυπνες Κατοικίες» προσφέρουν στους ενοίκους πάρα πολλές διευκολύνσεις και συμβάλλουν στην εξοικονόμηση πολύτιμου χρόνου. Οι παρεχόμενες διευκολύνσεις πολλαπλασιάζονται καθώς, εκτός από τις βασικές λειτουργίες, δίνεται επίσης η δυνατότητα στον ιδιοκτήτη να προγραμματίσει το σύστημα και να δημιουργήσει δικά του σενάρια, προκειμένου να καλύψει πλήρως τις δικές του εξατομικευμένες ανάγκες. Ορισμένα από τα σενάρια που μπορούν να εφαρμοστούν όσον αφορά τις συνήθεις λειτουργίες των έξυπνων σπιτιών, παρουσιάζονται ενδεικτικά παρακάτω [23].

- Έλεγχος ενέργειας, θέρμανσης, αερισμού, νερού
  Παρακολούθηση κατανάλωσης ενέργειας και νερού ώστε να ληφθούν οι κατάλληλες συμβουλές για τη μείωση των πόρων της ενέργειας.
- Φωτισμός: Αυτοματοποιημένος φωτισμός που αυξομειώνεται κατά τη διάρκεια της μέρας, προκατασκευασμένα σενάρια φωτισμού, αυτόματη ενεργοποίηση και απενεργοποίηση των φώτων κτλ.
- Ασφάλεια
  - Συστήματα ανίχνευσης εισβολής

  Ανίχνευση των παραθύρων και των θυρών ώστε να γίνονται αντιληπτές οι παραβιάσεις από εισβολείς.
  - Συστήματα ανίχνευσης βλάβης

  Ανίχνευση πυρκαγιάς, πλημμύρας, προστασία από βραχυκυκλώματα.

Οι φορείς που εμπλέκονται σ' αυτά τα σενάρια αποτελούν μια πολύ ετερογενή ομάδα. Διάφοροι φορείς συνεργάζονται στο σπίτι του κάθε χρήστη, όπως εταιρείες του Διαδικτύου, κατασκευαστές συσκευών, πάροχοι υπηρεσιών οπτικοακουστικών μέσων, εταιρίες κοινής ωφέλειας ηλεκτρικής ενέργειας κ.α.

## 3.1 Εφαρμογές Αυτόματων Συστημάτων

Σε διεθνές επίπεδο έχουν αναπτυχθεί διάφορα συστήματα και πρωτόκολλα επικοινωνίας για την υλοποίηση των τεχνολογιών, εστιάζοντας στο βαθμό αυτοματοποίησης και την ενεργειακή διαχείριση μιας έξυπνης οικίας [24] [25].

Στην Αμερική, επικρατούν οι τεχνολογίες PLC (Power Line Carrier) με το X10 πρώτο στη λίστα, μία τεχνολογία που δεν απαιτεί καμία επιπρόσθετη καλωδίωση του κτηρίου, καθιστώντας ικανή την ηλεκτρική εγκατάσταση σε υπάρχοντα κτήρια και σπίτια. Η επικοινωνία ανάμεσα στους διάφορους πομπούς και δέκτες πραγματοποιείται με την αποστολή και λήψη δεδομένων που γίνονται μέσα από την υπάρχουσα καλωδίωση, αξιοποιώντας τους ρευματοφόρους αγωγούς της κατοικίας.

Στην Ευρώπη συστήματα αυτομάτου ελέγχου όπως το Instabus KNX της Siemens, το Smart - House της Carlo Gavazzi, το C-bus της Clipsal είναι τεχνολογίες που βασίζονται σε ανεξάρτητη καλωδίωση, το λεγόμενο σύστημα bus. Κεντρικό σημείο των συστημάτων είναι η δημιουργία ενός αποκεντρωμένου συστήματος μεταφοράς και επεξεργασίας δεδομένων.

### Τι είναι ένα Bus Σύστημα;

Δημιουργήθηκε στο πλαίσιο μιας Ευρωπαϊκής συνεργασίας, από όπου προέκυψε το European Installation Bus (EIB). Με τη νέα αυτή τεχνολογία δίνεται η δυνατότητα ελέγχου και παρακολούθησης όλων των φορτίων μέσω ελεγχόμενων συσκευών (controllers) που συνδέονται σε μια κοινή γραμμή, η οποία ονομάζεται BUS (δίαυλος επικοινωνίας).

Χαρακτηριστικό αποτελεί η εύκολη σύνδεση των controllers στο bus καθώς χρειάζεται μόνο ένα ζεύγος αγωγών από το οποίο γίνεται η τροφοδοσία αλλά και η μεταφορά των δεδομένων. Το σύστημα bus δημιουργεί ένα σταθερό αξιόπιστο σύστημα αυτοματισμού για οικίες και κτήρια, καθιστώντας το ιδανικό τόσο σε επαγγελματικές κτηριακές εγκαταστάσεις όσο και σε οικιακές υποδομές.

## Σύστημα “Smart – House”

Η νέα σειρά προϊόντων bus της Carlo Gavazzi εστιάζει σε εφαρμογές Home Automation [26]. Το «Smart House» είναι ένα αποκεντρωμένο σύστημα αυτοματισμού που απλοποιεί τη χρήση όλου του τεχνολογικού εξοπλισμού μιας οικίας, παρέχοντας δυνατότητες ελέγχου και επίβλεψης φωτισμού, θέρμανσης, κλιματισμού και ασφάλειας. Αυτή η πρωτοποριακή δομή ανοίγει νέους ορίζοντες στη μείωση της ενεργειακής κατανάλωσης και αυξάνει την εύχρηστη διαχείριση των οικιακών συσκευών λόγω της συνολικής εποπτείας από οπουδήποτε.

Η χρήση των ευφυών bus τροφοδοτούμενων μονάδων όπως: διακόπτες φωτισμού, ανιχνευτές κίνησης, αισθητήρες θερμοκρασίας και αποκεντρωμένοι ηλεκτρονόμοι, κάνουν την εγκατάσταση εύκολη και ευέλικτη, γιατί δεν απαιτείται πλέον η χρήση μεγάλου όγκου καλωδίων προς τον κεντρικό πίνακα. Όλες οι μονάδες χρησιμοποιούν το ίδιο δισύρματο καλώδιο για τη σύνδεση με έναν Κεντρικό Ελεγκτή Smart House.

Ο Κεντρικός Ελεγκτής Smart House (SH Controller) προσφέρει μία συλλογή από προκαθορισμένες λειτουργίες που απλά χρειάζονται να παραμετροποιηθούν. Για παράδειγμα, η λειτουργία “Σενάριο” επιτρέπει σε μία είσοδο να ενεργοποιήσει ταυτόχρονα πολλές εξόδους ενώ η λειτουργία “Πραγματικός Χρόνος” επιτρέπει τη λειτουργία των εξόδων σε συγκεκριμένες ώρες και ημέρες της εβδομάδας. Ο έλεγχος θερμοκρασίας, συστήματος ασφαλείας και σεναρίων φωτισμού είναι επίσης παραδείγματα προκαθορισμένων λειτουργιών, που κάνουν το προγραμματισμό εύκολο.

Ο SH Controller στηρίζεται στο λειτουργικό σύστημα Windows CE, δίνει ιδιαίτερη βαρύτητα στην δικτυακή επικοινωνία, έχει ενσωματωμένο Web Server για επικοινωνία με Smart Phones, ενσωματωμένο φορτιστή μπαταριών, ενσωματωμένο πρωτόκολλο επικοινωνίας Modbus RS 232 και ModbusTCP, και μεγάλες δυνατότητες επέκτασης.

## Το σύστημα INSTABUS EIB/KNX

Η κατασκευή έξυπνων κτηρίων είναι πλέον εφικτή με τις πολλαπλές δυνατότητες που προσφέρει η νέα ευρωπαϊκή τεχνική εγκαταστάσεων KNX, όπως πλέον ονομάζεται σήμερα ως παγκόσμιο πρωτόκολλο επικοινωνίας.

Το instabus EIB/KNX της Siemens [27] είναι ένα νέο ευρωπαϊκό αποκεντρωμένο σύστημα μεταφοράς και επεξεργασίας δεδομένων για την ευέλικτη διαχείριση των λειτουργιών οι οποίες αφορούν μια ηλεκτρική εγκατάσταση κτηρίου ειδικής ή γενικής χρήσης. Σε ένα δίκτυο-Bus συνδέονται όλα τα ενεργά μέρη του συστήματος όπως αισθητήρες (διακόπτες, αισθητήρια φωτός, αισθητήρια θερμοκρασίας, ανιχνευτές κίνησης) και εντολείς ή έξοδοι (δυαδικές έξοδοι, ηλεκτρονόμοι, ρυθμιστές κλπ).

Όλες αυτές οι συσκευές προγραμματίζονται, αποκτούν λογική και ευφυΐα και ονομάζονται συνδρομητές του δικτύου. Η διασύνδεση των συνδρομητών γίνεται με ένα διπολικό καλώδιο. Το καλώδιο μπορεί να είναι και ένα τηλεφωνικό (τύπου YCYM 2x2x0,8mm2), όπου το ελεύθερο ζεύγος μπορεί να παραμείνει σαν εφεδρικό. Το καλώδιο αυτό μεταφέρει τις πληροφορίες και ταυτόχρονα τροφοδοτεί και τους συνδρομητές με την απαραίτητη τάση λειτουργίας 24V DC συνδέοντάς τους παράλληλα. Οι γραμμές ισχύος (230/400V) οδεύουν από τον πίνακα διανομής στους εντολείς και από εκεί στις καταναλώσεις.

Το σύστημα της Siemens instabus KNX υπάρχουν συσκευές για εξωτερική χρήση καθώς και για εσωτερική τοποθέτηση σε ηλεκτρολογικό πίνακα. Μπορεί να εγκατασταθεί σε επαγγελματικά κτήρια και κατοικίες ενσωματώνοντας σε ένα ενιαίο περιβάλλον ελέγχου και έξυπνης διαχείρισης των διάφορων ανεξάρτητων συστημάτων (έλεγχος φωτισμού, θέρμανσης, έλεγχος καταναλισκόμενης ενέργειας και φορτίων, απομακρυσμένοι χειρισμοί κ.τ.λ).

## 3.2 Έξυπνες Υλοποιήσεις του IoT στην Ελλάδα

Υπάρχουν και άλλες εταιρείες που έχουν ασχοληθεί με αυτοματισμούς συνδυάζοντας τεχνολογίες IoT χαράζοντας τον δικό τους δρόμο σε τεχνογνωσία και βρίσκουν εφαρμογή στον τομέα του έξυπνου σπιτιού.  που συναντάμε στην Ελλάδα (όπως η Watt and Volt).

**D-link**

Η D-LINK [28] είναι μία εταιρεία η οποία άρχισε να διεισδύει στον χώρο του διαδικτύου των αντικειμένων εστιάζοντας στις εφαρμογές του τομέα του έξυπνου σπιτιού. Ανέπτυξε τη δική της εφαρμογή για την χρήση των έξυπνων συσκευών της, την mydlinkhome app. Βασικό στοιχείο του σχεδιασμού αποτελεί ένα κεντρικό hub,

δηλαδή μία κεντρική μονάδα ελέγχου, η οποία αναλαμβάνει να διαχειριστεί ασύρματα τις επιμέρους συσκευές-ελεγκτές (που παρέχονται από την εταιρεία όπως έξυπνες πρίζες, αισθητήρες κίνησης κλπ) για την αυτοματοποίηση και τον έλεγχο των καταστάσεων που μπορούν να προκύψουν σε μία κατοικία.

Επιπλέον διαχειρίζεται το σύστημα ασφάλειας του σπιτιού, το οποίο αποτελείται από αισθητήρες και κάμερες για την παροχή ασφάλειας, με δυνατότητα πρόσβασης στο χρήστη από οπουδήποτε. Σημαντική είναι επίσης η δυνατότητα της ασύρματης μετάδοσης ήχου που παρέχει το mydlink Home music everywhere, το οποίο μέσω του DLNA[1] συνδέεται με οποιαδήποτε συσκευή ήχου. Η συγκεκριμένη συσκευή λειτουργεί και σαν ενισχυτής σήματος.

Ένα μειονέκτημα είναι ο περιορισμός των αισθητήρων και των περιφερειακών μονάδων που θέτει εμπόδια στην υλοποίηση μιας ολοκληρωμένη λύσης που θα ανταποκρίνεται στις απαιτήσεις μιας έξυπνης κατοικίας. Μια απουσία είναι αυτή του έξυπνου ηλεκτρονόμου, ο οποίος θα μπορούσε να ενταχθεί στον κεντρικό ηλεκτρικό πίνακα του σπιτιού για τον έλεγχο συσκευών μεγάλης ισχύος για παράδειγμα ενός οικιακού θερμοσίφωνα.

**SmartWatt**

Μία πρωτοποριακή υλοποίηση στον τομέα του έξυπνου σπιτιού αποτελεί η εφαρμογή smartWatt [29] της εταιρείας Watt&Volt, που δραστηριοποιείται στο χώρο της παροχής ενέργειας. Πρόκειται για ένα σύστημα ενεργειακής παρακολούθησης που προσφέρει διαρκή έλεγχο των συσκευών σε πραγματικό χρόνο με απώτερο στόχο τη βελτιστοποίηση του ενεργειακού κόστους από πλευράς οικονομικού ελέγχου και ενεργειακής διαχείρισης.

Οι έξυπνες συσκευές που προστίθενται (έξυπνες λάμπες, αισθητήρες, έξυπνοι ηλεκτρονόμοι, έξυπνες πρίζες για τον έλεγχο οποιασδήποτε ηλεκτρικής συσκευής) είναι ευέλικτες και εύκολες στην εγκατάσταση, με δυνατότητα επανατοποθέτησης όπου παρουσιαστεί ανάγκη, χωρίς καλώδια και επεμβάσεις στην ηλεκτρολογική εγκατάσταση. Επικοινωνούν ασύρματα μέσω του πρωτοκόλλου ZigBee με ένα κεντρικό Gateway, δηλαδή έναν κεντρικό δίαυλο επικοινωνίας. Το Gateway

[1] DLNA (Digital Living Network Alliance) Ένα πρότυπο, μια τεχνολογία ανταλλαγής ψηφιακών δεδομένων μεταξύ διαφορετικών ηλεκτρικών συσκευών, ιδρύθηκε το 2003 από μία ομάδα εταιρειών με τη Sony στον πρωταγωνιστικό ρόλο

συνδέεται απευθείας με το δρομολογητή (router) της διαδικτυακής σύνδεσης, δίνοντας έτσι στους χρήστες τη δυνατότητα πρόσβασης από οπουδήποτε μέσω της πλατφόρμας smartWatt. Η πλατφόρμα παρέχει ένα εύχρηστο περιβάλλον συμβατό και σε κινητά τηλέφωνα android/IOS, καταγράφοντας και αποτυπώνοντας διαγραμματικά όλες τις παραμέτρους της ενεργειακής κατανάλωσης.

## 3.3 Σύστημα Παρούσας Εργασίας

### 3.3.1 Μεθοδολογία

Στο πλαίσιο της ανάπτυξης μεθόδων για την αποδοτική ενεργειακή διαχείριση σε μια οικία, μελετήθηκαν διάφορα συστήματα αυτοματισμού και IoT εφαρμογών, εστιάζοντας στις υλοποιήσεις τους στον ενεργειακό τομέα, με στόχο να προχωρήσουμε στη δική μας ανάπτυξη ενός συστήματος εκτίμησης της κατανάλωσης ενέργειας [30] [31] [32].

Τα παραπάνω συστήματα (που αναφέρθηκαν στην προηγούμενη ενότητα) αποτελούν εμπορικά προϊόντα τα οποία υπό προϋποθέσεις μπορούν να συντελέσουν σημαντικές υλοποιήσεις στη διαχείριση ενέργειας. Ένα σημαντικό μειονέκτημα ωστόσο αποτελεί η εξάρτηση του κάθε συστήματος από τον συγκεκριμένο εξοπλισμό της αντίστοιχης εταιρείας, με αποτέλεσμα να μην υπάρχει μια ομαλή και διαφανής επικοινωνία με άλλα συστήματα.

**Ανάπτυξη του παρόντος Συστήματος**

Η έρευνά μας εστιάστηκε στην ιδέα μιας υλοποίησης ενός ανοιχτού συστήματος με στόχο μελλοντικές ερευνητικές εργασίες να μπορούν να το αναβαθμίσουν καθώς και να το επεκτείνουν. Το γεγονός ότι το όλο εγχείρημα πρόκειται για ένα open source έργο ανοιχτού κώδικα προς όλους, το καθιστά επεκτάσιμο και του αποδίδει σημαντικές προοπτικές εξέλιξης, χωρίς να έχει κερδοσκοπικό σκοπό, αλλά στοχεύοντας σε καθολική χρήση για την εξοικονόμηση ενέργειας. Επιπλέον στόχος της ανάπτυξης ήταν να μην υπάρχει περιορισμός από τη χρήση ενός συγκεκριμένου, εξειδικευμένου μικρο-υπολογιστικού συστήματος, αλλά αντιθέτως να υπάρχει μια πληθώρα επιλογών σε επίπεδο υλικού με κριτήριο το κόστος και την ποιότητα που θα παρουσιάζει πλήρη συμβατότητα όσον αφορά τα προγράμματα λογισμικού.

Στο πρακτικό μέρος η υλοποίησή μας βασίστηκε σε τέσσερα στάδια:

- Το πρώτο στάδιο περιείχε πειραματικές υλοποιήσεις για την εξαγωγή δεδομένων και της ανάλυσής τους σε επίπεδο ποιότητας και ακρίβειας με απώτερο σκοπό την επιλογή των υλικών που θα χρησιμοποιηθούν στο τελικό σύστημα.
- Στο δεύτερο στάδιο η έρευνά μας επικεντρώθηκε στην αρχιτεκτονική του συστήματος και στις τεχνολογίες που θα χρησιμοποιηθούν για την επικοινωνία των επιμέρους συστημάτων, και κατ' επέκταση την ασφαλή μετάδοση των δεδομένων και την αδιάλειπτη λειτουργία τους σε πραγματικές συνθήκες δικτύωσης. Η μελέτη των παραπάνω σε αυτό το στάδιο ανέπτυξε την ιδέα για ένα σύστημα που θα είναι κεντροποιημένο στη βάση μιας πλατφόρμας και θα μπορεί να εξυπηρετήσει πολλούς χρήστες με πολλά συστήματα αυτοματισμού [33].
- Το τρίτο στάδιο περιλαμβάνει την υλοποίηση της εφαρμογής συστήματος η οποία είχε ως βάση τη δυνατότητα επέκτασης των μονάδων ελέγχου. Επιλέχθηκε το αρχιτεκτονικό πρότυπο πάνω στο οποίο βασίστηκε η πλατφόρμα αλληλεπίδρασης του χρήστη με το σύστημα και δόθηκε έμφαση στη συλλογή και καταγραφή των πληροφοριών για τη δημιουργία στατιστικών γραφημάτων και την ενημέρωση του κάθε χρήστη για την κατανάλωση των επιλεγμένων συσκευών.
- Στο τελευταίο στάδιο συγκεντρώθηκε η μελέτη των επεκτάσεων για ένα σύστημα εφαρμόσιμο σε πραγματικές συνθήκες. Ερευνήθηκαν εκτιμήσεις και προβλέψεις που μπορεί να προκύψουν από την ανάλυση των ενεργειακών στατιστικών δεδομένων για τη δημιουργία ενός έξυπνου συστήματος που θα έχει τη δυνατότητα λήψης αποφάσεων και παρουσίασης προτάσεων για τη βέλτιστη ενεργειακή απόδοση.

**Συγκριτική Ανάλυση**

Παρακάτω παρατίθεται ένας συγκριτικός πίνακας που προκύπτει από την ανάλυση των χαρακτηριστικών μεταξύ των IoT εφαρμογών της Ελλάδας που παρατέθηκαν και του δικού μας αναπτυξιακού συστήματος. Θα πρέπει να σημειωθεί ότι οι εφαρμογές που μελετήθηκαν αποτελούν συστήματα γενικής διαχείρισης αυτοματισμών σπιτιού και ότι η σύγκρισή μας εστιάστηκε στο κομμάτι της ενεργειακής διαχείρισης.

**Πίνακας 3-1: Συγκριτικός Πίνακας Συστημάτων**

| | Watt&Volt (SmartWatt) | D-link | Πειραματικό Σύστημα |
|---|---|---|---|
| Προσανατολισμός | Επιχειρηματικός | Επιχειρηματικός | Ακαδημαϊκός |
| Χρέωση Υπηρεσιών | Περιλαμβάνεται | Περιλαμβάνεται | Δεν περιλαμβάνεται |
| Κόστος Συστήματος | Υψηλό | Υψηλό | Χαμηλό |
| Άδειες Λογισμικού | Κλειστός Κώδικας | Κλειστός Κώδικας | Ελεύθερο Λογισμικό |
| Συντήρηση | Πολιτική Εταιρείας | Πολιτική Εταιρείας | Ευρύ Φάσμα Επιλογών |
| Διαθεσιμότητα συνδεδεμένων ελεγκτών | Μεγάλη | Μέτρια | Στάδιο Ανάπτυξης |
| Ευκολία στη χρήση | Πολύ καλή | Καλή | Καλή |
| Ποιότητα Γραφημάτων/ Ανάλυση Δεδομένων | Πολύ Καλή | Καλή | Καλή (με δυνατότητες περαιτέρω ανάπτυξης ) |
| Χρονοπρογραμματισμός | Σενάρια Ενεργοποίησης / Απενεργοποίησης | Σενάρια Ενεργοποίησης / Απενεργοποίησης | Σενάρια Ενεργ. / Απενεργ. με δυνατότητες ομαδοποίησης των συσκευών |
| Προβλέψεις Καταναλώσεων | Παρέχεται | - | Στάδιο Ανάπτυξης |
| Συνεργασία / Επικοινωνία | Με συστήματα της ίδιας εταιρείας | Με συστήματα της ίδιας εταιρείας | Χωρίς περιορισμούς |

Στα παρακάτω κεφάλαια αναλύεται η υλοποίηση όλων των τμημάτων του συστήματος σε βήματα, δίνοντας έμφαση στα σημεία που έγιναν προγραμματιστικές επιλογές, προκειμένου να δοθούν πληροφορίες που θα συμβάλλουν σε περαιτέρω εξερεύνηση του παρόντος έργου.

# 4 Τεχνολογίες Υλοποίησης

Σε αυτήν την ενότητα προσδιορίζονται οι πλατφόρμες ανάπτυξης λογισμικού, οι τεχνολογίες και τα υπόλοιπα εργαλεία που χρησιμοποιήθηκαν, και αναλύεται η αρχιτεκτονική στην οποία βασίστηκε το υλικό μέρος της διπλωματικής εργασίας.

## 4.1 Λογισμικό Πακέτο

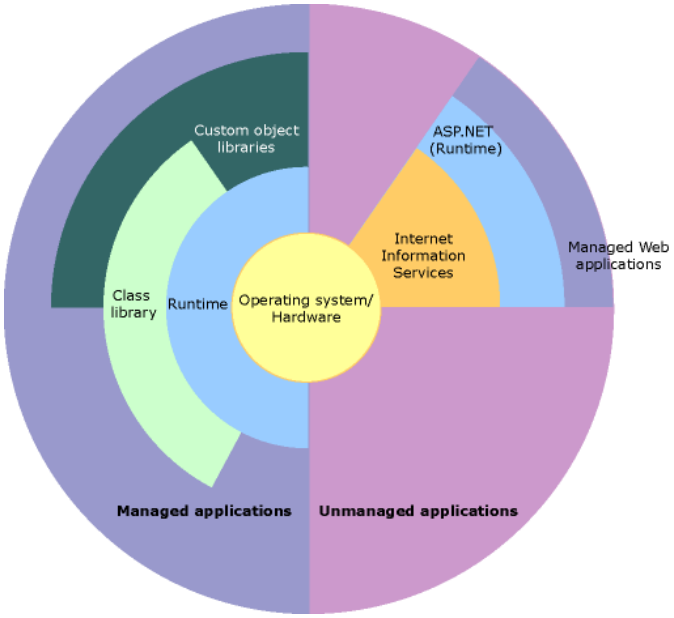
### 4.1.1 Η γλώσσα προγραμματισμού c# και το περιβάλλον .NET

Η C# (CSharp) [34] είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού η οποία αναπτύχθηκε από μια ομάδα ανάπτυξης λογισμικού της Microsoft με επικεφαλής τον Anders Hejlsberg, τον δημιουργό της Delphi, στα πλαίσια της προώθησης του περιβάλλοντος .NET. Η C # εισάγει μερικά μοναδικά και ισχυρά χαρακτηριστικά, όπως delegates (που μπορούν να θεωρηθούν ως ασφαλείς τύπου δείκτες συναρτήσεων), εκφράσεις λάμδα που εισάγουν τα στοιχεία των λειτουργικών γλωσσών προγραμματισμού, καθώς επίσης και ένα απλούστερο ενιαίο ταξικό μοντέλο κληρονομικότητας (από τη C ++). Είναι object-oriented, έρχεται με μια εκτεταμένη class library, και υποστηρίζει χειρισμό των exception, πολλαπλούς τύπους πολυμορφισμού και το διαχωρισμό των διεπαφών από την υλοποίηση. Τα χαρακτηριστικά αυτά, σε συνδυασμό με τα ισχυρά εργαλεία ανάπτυξης, την πλατφόρμα υποστήριξης, και τα generics, καθιστούν τη C # μια καλή επιλογή για πολλούς τύπους έργων ανάπτυξης λογισμικού, όπως έργα ταχείας ανάπτυξης εφαρμογών, εφαρμογές Διαδικτύου και έργα με αυστηρές απαιτήσεις αξιοπιστίας. Testing frameworks, όπως το NUnit κάνουν τη C # κατάλληλη για test - driven development.

Η C# αναπτύχθηκε ώστε να υποστηρίζει τις δυνατότητες του περιβάλλοντος .NET. Το .NET αποτελείται από το common language run time (CLR) και το .NET class library.

- The CLR είναι ένα run-time περιβάλλον το οποίο εκτελεί τον κώδικα και παρέχει υπηρεσίες που κάνουν την διαδικασία παραγωγής κώδικα ευκολότερη.
- Το .NET class library είναι μία βιβλιοθήκη από κλάσεις , interfaces, και άλλου τύπους που παρέχουν λειτουργικότητα στο σύστημα

Ο κώδικας που εκτελείται στο CLR έρχεται υπό μορφή assemblies. Το .NET χαρακτηρίζεται από την λεγόμενη just in time μεταγλώττιση. Ο κώδικας στη γλώσσα προγραμματισμού που χρησιμοποιεί ο χρήστης μεταγλωττίζεται αρχικά σε κάποια ενδιάμεση γλώσσα (IL) η οποία αποθηκεύεται σε ένα εκτελέσιμο αρχείο ή σε μια βιβλιοθήκη. Όταν ο χρήστης τρέξει το πρόγραμμα το CLR διαβάζει το IL(intermediate language) το μεταγλωττίζει σε κώδικα windows και στην συνέχεια τον εκτελεί.




**Εικόνα 4-1: Η σχέση του CLR και του class library**

Η χρήση του περιβάλλοντος .NET απλοποιεί αρκετές από τις διαδικασίες ανάπτυξης εφαρμογών και παρέχει ένα περιβάλλον εκτέλεσης κώδικα που προωθεί την ασφαλή εκτέλεση, συμπεριλαμβανομένου κώδικα από αγνώστου ή ημι-εμπιστεύσιμης πηγής (third party libraries) [35].

### 4.1.2 ASP.NET

Το Asp.net[2] είναι ένα προγραμματιστικό περιβάλλον της Microsoft που αναπτύχθηκε για διαδικτυακό προγραμματισμό προκειμένου να δώσει τη δυνατότητα σε προγραμματιστές να δημιουργήσουν δυναμικές ιστοσελίδες, διαδικτυακές εφαρμογές και διαδικτυακές υπηρεσίες.

[2] Επίσημη ιστοσελίδα: https://www.asp.net

Με το ASP.NET, μπορούν να δημιουργηθούν γρήγορα web sites που βασίζονται σε HTML, CSS και JavaScript, προσθέτοντας στη συνέχεια και πιο σύνθετες δυνατότητες όπως APIs Web, Φόρμες πάνω σε δεδομένα ή και συναλλαγές σε πραγματικό χρόνο.

### 4.1.3 ASP.NET MVC

Το ASP.NET MVC είναι ένα framework για τη δημιουργία εφαρμογών διαδικτύου που αναπτύχθηκε από τη Microsoft και υλοποιεί το πρότυπο MVC (Model View Controller). Πρόκειται για ένα ισχυρό και ελαφρύ εργαλείο για την οικοδόμηση εξελιγμένων ιστοσελίδων που επιτρέπει σε έναν καθαρό διαχωρισμό των εργασιών (λογική προσπέλασης δεδομένων και λογική παρουσίασης) και δίνει τον πλήρη έλεγχο σήμανσης για ευέλικτη και γρήγορη ανάπτυξη.

Είναι ένας τρόπος διαχωρισμού λογικών σε μία εφαρμογή παραδείγματος λογικής προσπέλασης δεδομένων και λογικής παρουσίασης. Ο διαχωρισμός αυτός προσθέτει μία επιπλέον πολυπλοκότητα, αλλά τα εξαιρετικά οφέλη υπερκαλύπτουν την επιπλέον προσπάθεια.

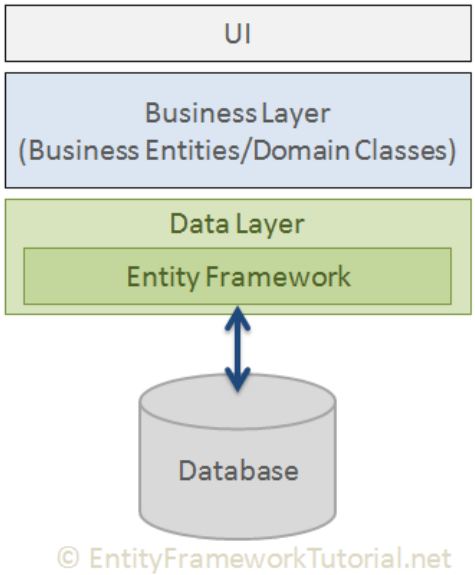
Βασισμένο σε ASP.NET, το MVC δίνει τη δυνατότητα οικοδόμησης μιας web εφαρμογής, ως σύνθεση τριών ρόλων καθένας από τους οποίους χειρίζεται ειδικές αναπτυξιακές πτυχές της αίτησης.

- Model (μοντέλο): ένα σετ από κλάσεις που περιγράφουν τα στοιχεία με τα οποία εργάζεται η εφαρμογή καθώς και τους επιχειρηματικούς κανόνες για το πώς τα δεδομένα μπορούν να αλλάξουν. Ως επί των πλείστον, αναλαμβάνει το κομμάτι της λογικής προσπέλασης των δεδομένων.
- View (όψη): ορίζεται το πώς πρέπει να είναι το περιβάλλον διεπαφής της εφαρμογής, η λογική της παρουσίασης
- Controller (ελεγκτής): ένα σετ από κλάσεις οι οποίες διαχειρίζονται την επικοινωνία του χρήστη, τις αλληλεπιδράσεις, την ενημέρωση του μοντέλου και ουσιαστικά τη συνολική ροή πληροφορίας της εφαρμογής

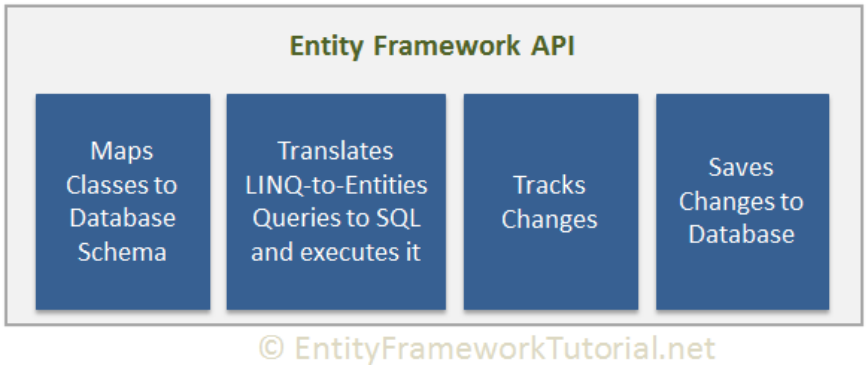
Το πλαίσιο ASP.NET MVC χρησιμοποιεί ένα μοτίβο πρόσοψης του ελεγκτή που επεξεργάζεται αιτήσεις εφαρμογών Web μέσω ενός ενιαίου ελεγκτή. Αυτό δίνει τη δυνατότητα να σχεδιαστεί μια εφαρμογή που υποστηρίζει μια πλούσια υποδομή δρομολόγησης.

### 4.1.4 Entity Framework

Το Entity Framework (EF) [36] είναι ένα ανοιχτού κώδικα Object/Relational Mapping (ORM) (αντικειμενοστρεφές) framework, το οποίο παρέχει μία χαρτογράφηση από το σχεσιακό σχήμα της βάσης δεδομένων σε αντικείμενα, υποστηρίζοντας την ανάπτυξη εφαρμογών λογισμικού δεδομένων. Αυτό το σύνολο τεχνολογιών επιτρέπει στους προγραμματιστές να δουλεύουν σε ένα πιο αφηρημένο επίπεδο, να χρησιμοποιήσουν σχεσιακά δεδομένα ως domain αντικείμενα, μειώνοντας έτσι την πολυπλοκότητα του κώδικα για την πρόσβαση στα δεδομένα της βάσης. Ένα τυπικό ORM εργαλείο όπως είναι το Entity, παράγει κλάσεις για τη διασύνδεση με τη βάση.




**Εικόνα 4-2: Η βασική ροή λειτουργίας του Entity Framework**




**Εικόνα 4-3: Το πλαίσιο εργασίας του Entity Framework**

Το Entity Framework επιτρέπει τη δημιουργία ενός μοντέλου παρέχοντας προγραμματιστικές προσεγγίσεις που χρησιμοποιούνται, είτε για τη στόχευση σε μία υπάρχουσα βάση, είτε για τη δημιουργία μιας καινούριας από την αρχή. Με τη μέθοδο Code First, που η ανάπτυξη στοχεύει στη συνεργασία με τη βάση, ορίζονται αρχικά τα μοντέλα μας, σχεδιάζοντας τις κλάσεις οντοτήτων με τη μορφή αντικειμένων και στη συνέχεια αυτόματα από αυτές τις κλάσεις, τα code-first APIS δημιουργούν τη βάση δεδομένων on the fly (αν δεν υπάρχει ήδη) και τους πίνακες, κάνοντας τις απαραίτητες αντιστοιχίσεις.

Οι προγραμματιστές με τη βοήθεια ερωτημάτων LINQ μπορούν να χειριστούν τα δεδομένα ως strongly typed αντικείμενα, δηλαδή τους επιτρέπεται να δημιουργήσουν εφαρμογές με πρόσβαση δεδομένων με χρήση εννοιολογικών μοντέλων αντί σχεσιακού σχήματος.

### 4.1.5 Linq

Το Linq (Language-Integrated Query) είναι μια τεχνολογία της Microsoft για την απλοποίηση και ενοποίηση της πρόσβασης σε δεδομένα (data access) από οποιαδήποτε πηγή δεδομένων. Με το Linq παρέχεται η δυνατότητα συγγραφής ευέλικτου κώδικα για την επικοινωνία με βάσεις και αρχεία, καθώς και για τη διαχείριση δομών δεδομένων.

Το Linq ενοποιεί τον τρόπο με τον οποίο τα δεδομένα μπορούν να αντληθούν από οποιοδήποτε αντικείμενο που υλοποιεί το interface IEnumerable<T> (δηλαδή οποιουδήποτε τύπου λίστα). Πιθανές πηγές δεδομένων μπορεί να είναι πίνακες, συλλογές, σχεσιακά δεδομένα, XML και Json. Τα κύρια τρία κομμάτια του Linq είναι τα παρακάτω:

- Linq σε αντικείμενα, που είναι ένα API που προσφέρει μεθόδους που ανακτούν δεδομένα από οποιοδήποτε αντικείμενο IEnumerable<T>. Αυτά τα queries γίνονται πάνω σε δεδομένα της μνήμης του προγράμματος.
- Linq στο ADO.NET, που προσφέρει κάποιες βασικές λειτουργίες από queries (Standard Query Operations) για σχεσιακά δεδομένα.
- Linq σε SQL, που χρησιμοποιείται για ερωτήματα σε σχεσιακές βάσεις δεδομένων όπως ο Microsoft Sql Server.

- Linq σε XML, που υποστηρίζει βασικές λειτουργίες ερωτημάτων πάνω σε δεδομένα που είναι σε μορφή XML.

Το μοντέλο Linq σε SQL παρέχει δυνατότητες object-relational mapping που επιτρέπουν στην εφαρμογή να χρησιμοποιήσει Language Integrated Query (LINQ), ώστε να επικοινωνεί με μια σχεσιακή βάση δεδομένων (μόνο με Transact-SQL). Χαρτογραφεί το μοντέλο αντικειμένου, το οποίο εκφράζεται με τον .NET Framework managed code, σε μια σχεσιακή βάση δεδομένων. Όταν η εφαρμογή τρέχει το LINQ σε SQL μεταφράζει language-integrated ερωτήματα σε Transact-SQL και στη συνέχεια στέλνει τα ερωτήματα στη βάση δεδομένων για εκτέλεση. Όταν η βάση δεδομένων επιστρέφει τα αποτελέσματα, το LINQ σε SQL μεταφράζει τα αποτελέσματα πίσω σε αντικείμενα, τα οποία μπορούν να διαχειριστούν στη γλώσσα προγραμματισμού που χρησιμοποιείται. Η εκτέλεση της LINQ συνδέεται πάντα με LAMBDA συναρτήσεις/εκφράσεις οι οποίες έχουν την έννοια των Annonymous Μεθόδων [37].

### 4.1.6 Front End Τεχνολογίες

#### *4.1.6.1 Javascript*

Η JavaScript[3] αποτελεί μια υψηλού επιπέδου, δυναμική μεταγλωττισμένη γλώσσα προγραμματισμού, καθώς οι περισσότερες εκτελέσεις εντολών γίνονται άμεσα χωρίς να έχει προηγηθεί μετάφραση του προγράμματος. Συχνά χρησιμοποιείται ως μέρος των φυλλομετρητών, των οποίων οι υλοποιήσεις επιτρέπουν αλληλεπίδραση με το χρήστη, για τον έλεγχό του προγράμματος περιήγησης, για την εφαρμογή ασύγχρονης επικοινωνίας και την τροποποίηση του περιεχομένου του εγγράφου που εμφανίζεται. Μπορεί και προσαρμόζει τις σελίδες, μέσω μηχανισμών ανίχνευσης, ανάλογα με τις απαιτήσεις των φυλλομετρητών. Πρόκειται για μία πρωτότυπη πολλαπλών παραδειγμάτων γλώσσα που υποστηρίζει αντικειμενοστρεφή και συναρτησιακό προγραμματισμό.

#### *4.1.6.2 JQUERY*

Η jQuery[4] είναι μια ανοιχτού κώδικα JavaScript βιβλιοθήκη, που στόχος της είναι να απλοποιήσει την υλοποίηση σεναρίων (scripting) στη πλευρά του πελάτη (client-side) της HTML. Πρόκειται για μία πολύ ισχυρή βιβλιοθήκη της JavaScript η οποία

[3] Επίσημη ιστοσελίδα: https://www.javascript.com
[4] Επίσημη ιστοσελίδα: http://jquery.com

χρησιμοποιείται σε πλήθος διαδικτυακών εφαρμογών καθώς προσφέρει αφαιρετικές ιδιότητες κατά τη συγγραφή των προγραμμάτων και είναι πλήρως επεκτάσιμη από άλλους προγραμματιστές.

#### *4.1.6.3 Bootstrap*

Το Bootstrap[5] είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος καθώς και προαιρετικές επεκτάσεις JavaScript. Είναι το πιο δημοφιλές πρόγραμμα στο GitHub και έχει χρησιμοποιηθεί από τη NASA και το MSNBC, μεταξύ άλλων. Βασικές πληροφορίες συμβατότητας των ιστοσελίδων ή εφαρμογές είναι διαθέσιμες για όλες τις συσκευές και τα προγράμματα περιήγησης.

## 4.2 Υλικό

Αρχικός σκοπός ήταν η απόκτηση του απαραίτητου υλικού για την υλοποίηση της ιδέας του συστήματος. Σ΄ αυτό το σημείο, λοιπόν, κρίθηκε απαραίτητο να οριστούν τα τεχνικά χαρακτηριστικά και οι προδιαγραφές των εξαρτημάτων, που χρησιμοποιήθηκαν στα διάφορα στάδια διεκπεραίωσης των πειραματικών διατάξεων.

### 4.2.1 Arduino Uno

Το Arduino [38] είναι μία υπολογιστική πλατφόρμα βασισμένη σε μία απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και αναπτυσσόμενο περιβάλλον για την εύκολη και απλή συγγραφή λογισμικού, που μπορεί να χρησιμοποιηθεί για την κατασκευή ψηφιακών συσκευών και διαδραστικών αντικειμένων.

Οι πλακέτες χρησιμοποιούν διάφορες εκδόσεις 8-bit Atmel AVR μικροελεγκτών ή 32-bit επεξεργαστών ARM Atmel. Αυτά τα συστήματα παρέχουν σύνολα ψηφιακών και αναλογικών ακροδεκτών I/O στους οποίους μπορούν να διασυνδεθούν διάφορες πλακέτες επέκτασης και άλλα κυκλώματα. Το Arduino διαθέτει σειριακό interface και ο μικροελεγκτής ATmega υποστηρίζει τη σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για τη μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για

[5] Επίσημη ιστοσελίδα: http://getbootstrap.com

αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.

Για τον προγραμματισμό των μικροελεγκτών, η πλατφόρμα Arduino παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) σύμφωνα με το πρόγραμμα Processing, το οποίο μπορεί να υποστηριχθεί από τις γλώσσες προγραμματισμού C και C++.

Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα «bootloader» δηλαδή ένα εσωτερικό κώδικα, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Το Arduino διατίθεται σε πολλές εκδόσεις ανάλογα με τις ανάγκες και τις δυνατότητες του κάθε προγραμματιστή. Γενικά όμως όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων «RS-232» και «TTL» [39].

**Επικοινωνία**

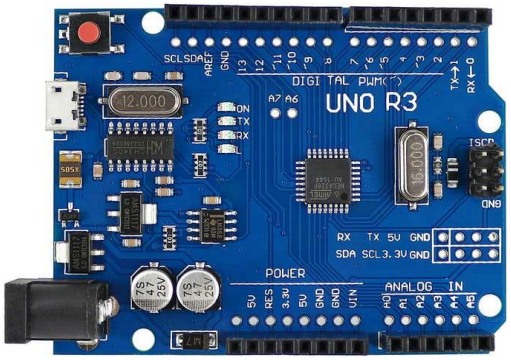
Το Arduino Uno μπορεί να αναπτύσσει επικοινωνία με έναν υπολογιστή, ένα άλλο Arduino, ή άλλους μικροελεγκτές. Ο μικροελεγκτής ATmega328 παρέχει σειριακή επικοινωνία με τη χρήση UART TTL (5V), η οποία είναι διαθέσιμη από τους ακροδέκτες 0 (λήψη RX) και 1 (εκπομπή TX).

Το Arduino με την σύνδεση του μέσω της θύρας USB, εμφανίζεται ως εικονική σειριακή θύρα COM στο λογισμικό του υπολογιστή, εφόσον ο μικροελεγκτής ATmega16U2 μετατρέπει τη σειριακή επικοινωνία σε μορφή USB. Επίσης, το Arduino περιλαμβάνει ένα τμηματικό όργανο ελέγχου το οποίο επιτρέπει την απλή μορφή κειμένου δεδομένων που αποστέλλονται προς και από την πλακέτα Arduino. Οι RX και TX λυχνίες LED στην πλακέτα θα αναβοσβήνουν όταν γίνεται μετάδοση δεδομένων μέσω της USB θύρας (USB- σειριακή σύνδεση με τον υπολογιστή).

**Arduino Uno Rev 3**

Όσον αφορά την αρχιτεκτονική του Arduino Uno Rev 3 [40], είναι μια πλατφόρμα βασισμένη στον ATmega 328 της Atmel. Περιέχει 14 ψηφιακούς ακροδέκτες εισόδου-εξόδου («I/O»), 6 αναλογικές εισόδους για την ανάγνωση αναλογικών τιμών από εξωτερικά στοιχεία (ποτενσιόμετρα, αναλογικοί αισθητήρες) και 6 ψηφιακές

εξόδους που μπορούν να χρησιμοποιηθούν ως PWM (Pulse Width Modulation), δηλαδή να έχουν τη δυνατότητα να παράγουν παλμούς με μεταβλητό πλάτος. Επιπλέον περιλαμβάνει έναν κεραμικό κρύσταλλο ταλαντωτή των 16 MHz. Η




**Εικόνα 4-4: Arduino Uno R3**

σύνδεση γίνεται με USB (Universal Serial Port) καλώδιο, ενώ υπάρχει και υποδοχή σύνδεσης με ρεύμα, καθώς και δυνατότητα εντός του κυκλώματος, σειριακού προγραμματισμού-ICSP («In-Circuit Serial Programming») και ένα κουμπί επανεκκίνησης (reset) σε περίπτωση που βραχυκυκλώσει η πλατφόρμα. Τα γενικά χαρακτηριστικά της πλατφόρμας παρουσιάζονται στον πίνακα.

**Πίνακας 4-1: Γενικά Χαρακτηριστικά της πλατφόρμας Arduino Uno Rev 3**

| | |
|---|---|
| Μικροελεγκτής | ATmega328 |
| Επεξεργαστής | 8-bit AVR |
| Τάση Λειτουργίας | 5V |
| Τάση Εισόδου (προτεινόμενη) | 7-12V |
| Τάση Εισόδου (ελάχιστη-μέγιστη) | 6-20V |
| Ψηφιακοί Είσοδοι/Εξοδοι | 14 (6 PWM έξοδοι) |
| Αναλογικές Είσοδοι | 6 |
| Συνεχές ρεύμα ανά ακροδέκτη εισόδο/εξόδου | 20mA |
| Συνεχές ρεύμα για τον ακροδέκτη 3.3V | 50mA |
| Μνήμη Flash | 32KB (ATmega328) από τα οποία τα 0.5 KB χρησιμοποιούνται για το σύστημα |
| Μνήμη SRAM | 2 KB (ATmega328) |

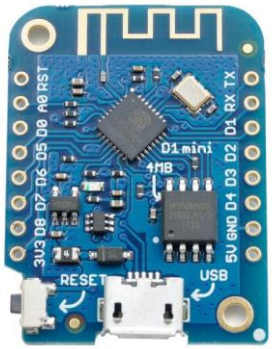
| | |
|---|---|
| EEPROM | 1 KB (ATmega328) |
| Ταχύτητα χρονισμού | 16 MHz |

### 4.2.2 Πρώτη σειρά μικροελεγκτών ESP8266

Η πρώτη σειρά από Wi-Fi μικροελεγκτές που δημιουργήθηκε από δυτικούς κατασκευαστές και συγκεκριμένα από την Al-Thinker, ονομάζεται ESP8266 και παραμένει ως σήμερα η πιο ευρέως διαθέσιμη. Αναφέρονται συλλογικά ως ESP-xx μονάδες (ESP-201, ESP-01, ESP-12F) και για να σχηματίσουν ένα λειτουργικό σύστημα ανάπτυξης, απαιτούν επιπλέον εξαρτήματα. Ειδικότερα χρειάζεται ένας TTL-to-USB προσαρμογέας (συνήθως ονομάζεται μία γέφυρα USB-to-UART) και ένα εξωτερικό τροφοδοτικό που αποδίδει τάση 3,3 Volt.

### 4.2.3 Wemos D1 Mini

Το wemos D1 mini [41] είναι ένας ESP-8266 μικροελεγκτής με δυνατότητες Wi-Fi. Αποτελεί μια πλακέτα που έχει ενσωματωμένα τη γέφυρα USB-to-UART (CH340) και ένα βύσμα micro-USB σε συνδυασμό με ένα ρυθμιστή ισχύος 3,3Volt για την




**Εικόνα 4-5: Wemos D1 Mini**

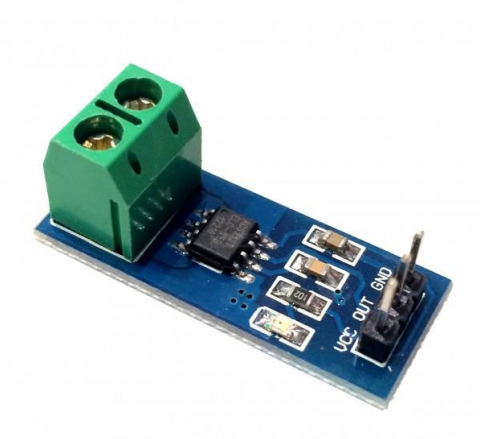
τροφοδοσία του. Η μονάδα αυτή ESP8266 σχηματίζει ένα σύστημα ανάπτυξης λειτουργικό, χωρίς να απαιτούνται επιπλέον εξαρτήματα με αποτέλεσμα να καθίσταται οικονομικότερη και ενεργειακά αποδοτικότερη αφού καταναλώνει μικρότερη ισχύ. Επιπλέον είναι συμβατό με το Arduino σε επίπεδο λογισμικού (υποστήριξη από Arduino IDE), παρέχει τη δυνατότητα δημιουργίας προγραμμάτων σε MicroPython καθώς και τη χρήση του firmware NodeMCU (Lua scipts).

**Πίνακας 4-2: Γενικά Χαρακτηριστικά της πλατφόρμας Wemos D1 Mini**

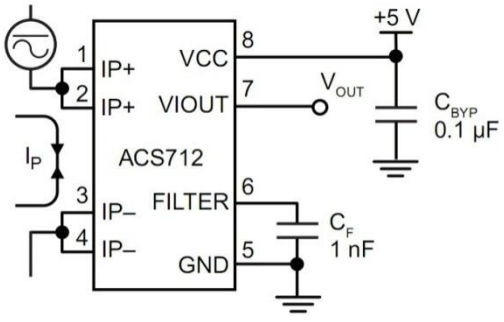
| | |
|---|---|
| Μικροελεγκτής | ESP-8266EX |
| Επεξεργαστής | 32-bit MCU |
| Τάση Λειτουργίας | 3.3V |
| Ψηφιακοί Είσοδοι/Εξοδοι | 11 (PWM / 12C / SPI/ 1-Wire) |
| Αναλογικές Είσοδοι | 1 (Μέγιστη τάση: 3.2V, ADC - 10bit) |
| Συνεχές μέγιστο ρεύμα ανά ακροδέκτες εισόδου/εξόδου | 12mA |
| Μνήμη Flash | 4M bytes |
| Ταχύτητα χρονισμού | 80 MHz / 160 MHz |
| Ασύρματη Σύνδεση | Wi-Fi 802.11 b/g/n |
| Μήκος | 34.2mm |
| Πλάτος | 25.6mm |

### 4.2.4 Αισθητήρας ACS712

Το ACS712 [42] είναι ένας οικονομικός αισθητήρας μέτρησης έντασης ρεύματος. Η λειτουργία του ολοκληρωμένου βασίζεται στην αρχή του φαινομένου Hall. Τροφοδοτείται από 5V (VCC) συνεχούς τάσης. Ως έξοδος παράγεται ένα σήμα τάσης το οποίο εξαρτάται γραμμικά από το ρεύμα που διαρρέει τους ακροδέκτες 1-4 του ολοκληρωμένου. Επιπλέον παρέχει απομόνωση με τους ακροδέκτες σήματος 5-8.




**Εικόνα 4-6: Acs712**




**Εικόνα 4-7: Κυκλωματική Διάταξη Acs712**

Η συγκεκριμένη συσκευή εκμεταλλεύεται το μαγνητικό πεδίο που δημιουργείται από έναν αγωγό, όταν αυτός διαρρέεται από ρεύμα. Χρησιμοποιώντας ένα μαγνητικό

αισθητήρα μετατρέπει την ένταση του μαγνητικού πεδίου σε μία αναλογική τάση. Η έξοδος του αισθητήρα είναι ρυθμισμένη στα 1/2 VCC ή περίπου στα 2.5V τάση η οποία αντιπροσωπεύει τη μηδενική ροή ρεύματος. Έτσι μια αρνητική ροή ρεύματος θα μας δώσει τάση κάτω από 2.5V και μια θετική ροή ρεύματος πάνω από 2.5V.

Σε περίπτωση που μετρούμε εναλλασσόμενο ρεύμα θα πρέπει η τάση εξόδου του αισθητήρα να μετατραπεί σε RMS τιμή.

### 4.2.5 Αισθητήρας SCT-013-030

Ο αισθητήρας ρεύματος που χρησιμοποιήθηκε σε αυτή την εργασία είναι ο μετασχηματιστής ρεύματος SCT-013-030 [43] της εταιρείας YHDC.

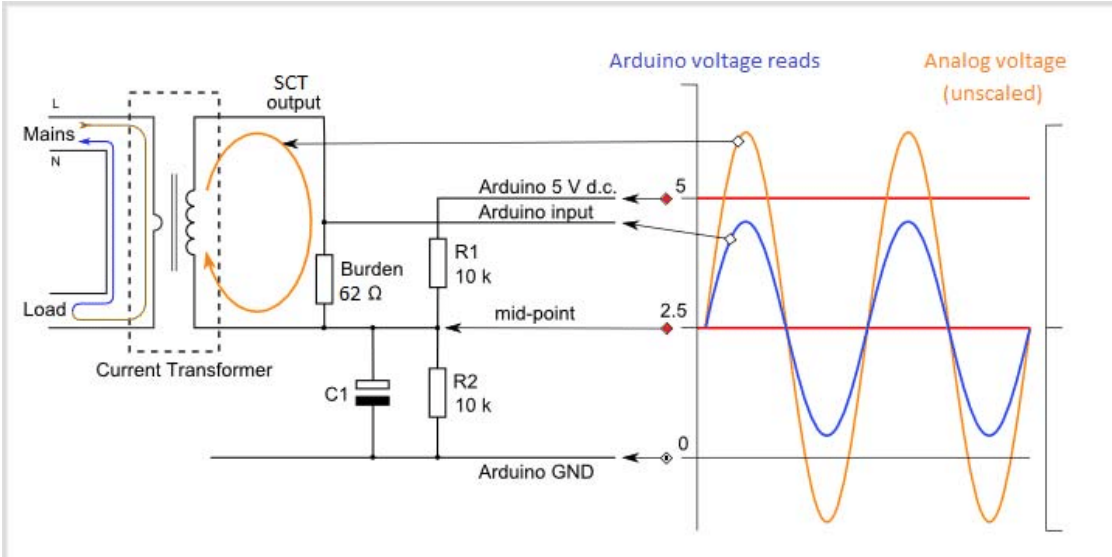
Πρόκειται για έναν μη -επεμβατικό (non-invasive) αισθητήρα ρεύματος που μπορεί να σταθεροποιηθεί γύρω από τη γραμμή παροχής ενός ηλεκτρικού φορτίου και να διαβάσει πόσο ρεύμα περνάει μέσα από αυτό. Ο πυρήνας του χωρίζεται και με αυτόν τον τρόπο ο αισθητήρας ανταποκρίνεται στο μαγνητικό πεδίο γύρω από τον αγωγό ρεύματος που περιβάλλει. Τέτοιου τύπου αισθητήρες είναι ιδανικοί για πανομοιότυπες εφαρμογές.


**Εικόνα 4-8: SCT-013-030**

Όπως κάθε μετασχηματιστής, έτσι και αυτός έχει το πρωτεύον τύλιγμα, έναν πυρήνα από φερρίτη και το δευτερεύον τύλιγμα. Το πρωτεύον πηνίο θα είναι η φάση ή ο ουδέτερος του καλωδίου που περνάει από τον κενό χώρο του αισθητήρα. Να σημειωθεί ότι δε θα πρέπει ο ουδέτερος με την φάση να περνούν μαζί μέσα από τον αισθητήρα, διότι σε αυτήν την περίπτωση θα είχαμε τη διαρροή ίσων ρευμάτων σε διαφορετικές όμως κατευθύνσεις, δηλαδή δύο ίσα και αντίθετα μαγνητικά πεδία που θα ακύρωναν το ένα το άλλο και επομένως δεν θα υπήρχε έξοδος στο δεύτερο πηνίο. Για κάθε 30 Ampere στο πρωτεύον πηνίο επάγει 1Volt στο δευτερεύον το οποίο αποτελείται από 1800 σπείρες. Η έξοδος που μας δίνει είναι από 0-1 Volt με

αντίστοιχες τιμές ρεύματος 0-30 Ampere που διαρρέει το πρωτεύον. Ο αισθητήρας ο συγκεκριμένος επιπλέον ενσωματώνει μια αντίσταση 62 Ohm.

Η έξοδος του μετασχηματιστή είναι μία σχεδόν ημιτονοειδής κυματομορφή η οποία έχει διακύμανση -1 +1 Volt που αντιστοιχεί στη μέγιστη τιμή μέτρησης του αισθητήρα (30 Ampere). Δεδομένου ότι όλοι οι μετατροπείς ADC μετρούν μόνο θετικές τάσεις, επιβάλλεται η θετική μετατόπισή της τάσης εξόδου για την εξάλειψη των αρνητικών τιμών. Για να καταστεί αυτό δυνατό υλοποιείται το παρακάτω κύκλωμα στην έξοδο του μετασχηματιστή:




**Εικόνα 4-9: Σχηματικό διάγραμμα κυκλώματος αντίστασης φορτίου και διαιρέτη τάσης**

Οι αντιστάσεις R1, R2 αποτελούν ένα διαιρέτη τάσης και προσθέτουν την πόλωση με τη δημιουργία μιας DC συνιστώσας 2.5V, ενώ ο πυκνωτής C1 παρέχει μία διαδρομή χαμηλής σύνθετης αντίστασης προς τη γείωση για το AC σήμα (εδώ επιλέχθηκε στα 10μF). Έτσι για να εξασφαλιστεί ότι το σήμα τάσης του μετασχηματιστή θα είναι μετρήσιμο από τον ADC μετατροπέα, επιλέγονται οι αντιστάσεις R1, R2 με ενδεικτική τιμή στα 10kΩ, όπως φαίνεται και στο κυκλωματικό διάγραμμα. Οι αντιστάσεις πρέπει να είναι σχετικά μεγάλων τιμών προκειμένου να έχουμε όσο το δυνατόν ελάχιστη διαρροή ρεύματος στο κύκλωμα, ιδιαίτερα σε περιπτώσεις που τα συστήματά μας τροφοδοτούνται από επαναφορτιζόμενους ηλεκτρικούς συσσωρευτές.

**Πίνακας 4-3: Τεχνικά Χαρακτηριστικά του αισθητήρα SCT-013-30**

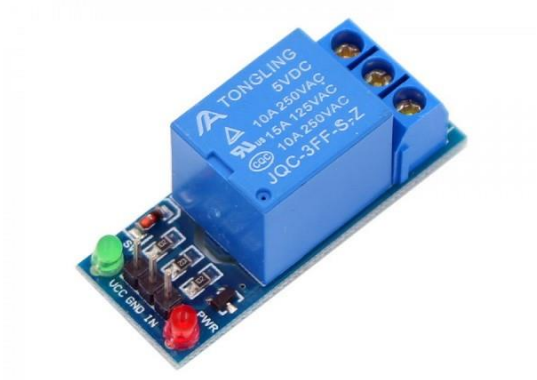
| Input current | Output voltage | non-linearity | Build-in sampling resistance ($R_L$) |
|---|---|---|---|
| 0-30A | o-1V | +-1% | 62Ω |
| Turn ratio | Resistance grade | Work temperature | Dielectric strength (between shell and output) |
| 1800: 1 | Grade B | -25°C ~ +70 °C | 1500V AC/1min 5mA |

### 4.2.6 Relay 5V-1 Channel Module Board

Το Relay module που χρησιμοποιήθηκε στο σύστημα για την διάταξη ελέγχου είναι το μοντέλο Relay 5V-1 Channel. Πρόκειται για μια διάταξη που ενσωματώνει έναν ηλεκτρονόμο και κάποια ενεργά στοιχεία τα οποία την καθιστούν ικανή να ελεγχθεί από έναν μικροελεγκτή. Περιλαμβάνει ακόμη δύο φωτοδιόδους για τις ενδείξεις λειτουργίας της και μπορεί να ελέγξει ηλεκτρικές συσκευές ισχύος μέχρι 2.2 KWatt περίπου.

**Πίνακας 4-4: Γενικά Χαρακτηριστικά Relay**

| | |
|---|---|
| **Μέγιστο Φορτίο** | AC 250V / 10A, DC 30V / 10A |
| **Τάση Λειτουργίας** | 5V |
| **Ρεύμα Λειτουργίας** | 15-20mA |
| **Ρεύμα Έλεγχου** | ≥ 4mA |
| **Οπτική Σήμανση** | 2 X φωτοδίοδοι |
| **Μέγεθος Μονάδας** | 4.7 x 2.9 x 1.8cm |




**Εικόνα 4-10: Διάταξη Relay**

**Πίνακας 4-5: Περιγραφή της διάταξης Relay**

| Επαφή | Περιγραφή |
|---|---|
| VCC | Επαφή τάσης 5V |
| GND | Επαφή Γείωσης |
| IN | Επαφή Ελέγχου Μονάδας τουλάχιστον 4mA |

# 5 Μετρητικές Διατάξεις και Διατάξεις Ελέγχου

## 5.1 Εισαγωγή

Για τις ανάγκες υλοποίησης των μετρητικών διατάξεων και τον έλεγχο της ηλεκτρικής κατανάλωσης των επιλεγμένων συσκευών εξετάστηκαν διάφορα υπολογιστικά συστήματα που κυκλοφορούν στην αγορά, με τη μορφή Singleboard microcontrollers (SBM).

Στα πρώτα στάδια της διπλωματικής δημιουργήθηκαν κάποιες πειραματικές κατασκευές οι οποίες στηρίχθηκαν στις δυνατότητες του μικροελεγκτή Arduino Uno [44]. Τα τυπικά αυτά μετρητικά κυκλώματα αποσκοπούσαν στη μέτρηση έντασης ρεύματος, για αυτό και τοποθετήθηκαν αντίστοιχου τύπου αισθητήρες (ACS712, SCT013-030), πολύ καλής ακρίβειας και χαμηλού κόστους. Βασικός στόχος αποτέλεσε η μελέτη και ο προσδιορισμός του ρόλου του κάθε συστήματος, μέσω μετρήσεων που διεξήχθησαν σε προκαθορισμένες, ίδιες εργαστηριακές συνθήκες, αναλύοντας τα συμπεράσματα, σε επίπεδο ακρίβειας αποτελεσμάτων και απόκρισης χρόνου.

**Διάταξη που θα δεχθεί τα ηλεκτρονικά κυκλώματα**

Αρχικά για λόγους προστασίας και για τη διευκόλυνση της διεξαγωγής των πειραμάτων αποφασίστηκε τα ηλεκτρονικά κυκλώματα να ενσωματωθούν σε ένα ηλεκτρολογικό επίτοιχο πίνακα μίας σειράς, με δυνατότητα χρήσης έως έξι στοιχείων (θέσεων) και με βαθμό στεγανότητας ip40[6].

Ενσωματώθηκαν μια πρίζα σούκο (ράγας) για την παροχή ενέργειας στις επιλεγμένες συσκευές κατανάλωσης καθώς και μια ενδεικτική λυχνία για την ένδειξη της τροφοδοσίας. Η λειτουργία του εν λόγω κυκλώματος ανέρχεται στα 16 Ampere εναλλασσόμενου ρεύματος, τάσης 230 Volt (οικιακό μονοφασικό ρεύμα) και κατά συνέπεια δίνεται η δυνατότητα της μέτρησης συσκευών με κατανάλωση περίπου έως 4000 Watt. Λόγου του περιορισμού της ισχύος που έχει ο ηλεκτρονόμος (10 Ampere), τα πειράματα μας περιορίστηκαν σε έλεγχο ισχύος μέχρι 2000Watt.

[6] Ο δείκτης στεγανότητας IP αποτελεί ένα διεθνή πρότυπο της IEC (International Electrotechnical Commision) που ταξινομεί κάθε ηλεκτρική συσκευή ως προς το βαθμό προστασίας της απέναντι στο νερό, τη σκόνη. Το πρώτο ψηφίο δηλώνει την Αντίσταση έναντι της Εισδοχής Στερεών Αντικειμένων και Σκόνης και παίρνει τιμές από 0 έως 6. Το δεύτερο ψηφίο δηλώνει την Αντίσταση έναντι της Εισροής Νερού και παίρνει τιμές από 0 έως 8.

## 5.2 Μετρητικές Διατάξεις

### 5.2.1 1η Πειραματική Διάταξη

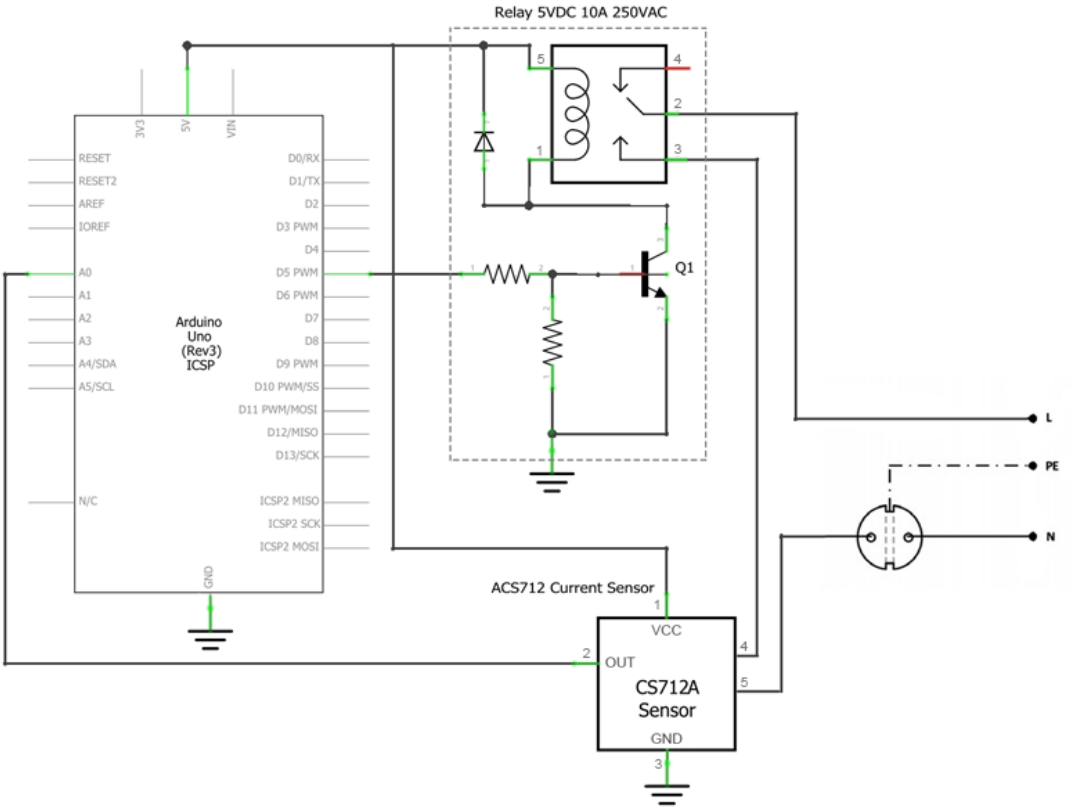
Η πρώτη μας μετρητική διάταξη περιλαμβάνει τα ακόλουθα:

- Πλατφόρμα μικροελεγκτή για λήψη και αποστολή των μετρήσεων καθώς και έλεγχο των αισθητήρων
- Αισθητήρες έντασης ρεύματος επαγωγικής λειτουργίας και κάποια εξαρτήματα απαραίτητα για τη σύνδεση των αισθητήρων στο μικροελεγκτή (αντιστάσεις, δίοδοι, πυκνωτές)
- Φορτίο το οποίο δημιουργεί τις συνθήκες μέτρησης της έντασης του ρεύματος που διαρρέει το κύκλωμα στο οποίο βρίσκονται οι αισθητήρες
- Λογισμικό εφαρμογής στον HY το οποίο ελέγχει την Πλατφόρμα μικροελεγκτή

Τα υλικά τα οποία χρησιμοποιήθηκαν στην κατασκευή και στον έλεγχο της λειτουργίας είναι:

- Πλατφόρμα μικροελεγκτή : Arduino UNO R3
- Αισθητήρας έντασης ρεύματος (πρότυπος): ASC712
- Ηλεκτρονόμος: 5V 1 channel Relay
- Φορτίο μεταβολής ισχύος: συσκευή στεγνώματος μαλλιών, αερόθερμο, καφετιέρα, αφυγραντήρας
- Λογισμικό εφαρμογής: Arduino Software (IDE)

Στο σχήμα που ακολουθεί παρουσιάζεται η σύνδεση της μετρητικής διάταξης η οποία είναι υπεύθυνη για τις μετρήσεις και τη συλλογή τους.




**Εικόνα 5-1: Συνδεσμολογία 1ης πειραματικής διάταξης**

Για τη διευκόλυνση της ανάλυσής του σχεδιαγράμματος θα θεωρήσουμε δύο νοητές πλευρές στο σχήμα.

Η μία πλευρά (L, Relay ακροδέκτες 2-3, CS712 ακροδέκτες 4-5, N) είναι αυτή που έρχεται σε επαφή με το εναλλασσόμενο μονοφασικό ρεύμα υψηλών εντάσεων (τάξης 220 Volt - 10 Ampere) και αποτελείται από τα στοιχεία: την πρίζα σούκο η οποία τροφοδοτεί τις προς μέτρηση ισχύος κατανάλωσης συσκευές, το module relay (250 Volt AC 10 Ampere) το οποίο αναλαμβάνει το ρόλο του διακόπτη ενεργοποίησης/απενεργοποίησης του εναλλασσόμενου κυκλώματος και τον αισθητήρα μέτρησης ηλεκτρικού ρεύματος. Τα προκείμενα στοιχεία είναι συνδεδεμένα εν σειρά στη διάταξη.

**Ενσωμάτωση του ηλεκτρονόμου στην κυκλωματική υλοποίηση**

Ο ηλεκτρονόμος που χρησιμοποιήθηκε αποτελεί ένα ενιαίο εξάρτημα (module) αλλά για τις ανάγκες της σχεδίασης, αναλύθηκε η διάταξη κατασκευής του. Ακολουθεί μια περιγραφική επεξήγηση του ηλεκτρονόμου.

Όπως παρουσιάζεται στο σχήμα συνδεσμολογίας, απαρτίζεται από τη μηχανική του διάταξη και από ένα σύνολο ηλεκτρονικών στοιχείων το οποίο συμμετέχει στη διαδικασία ελέγχου του [45].

Αναλυτικά έχουμε τα εξής στοιχεία:

- Ένα τρανζίστορ τύπου NPN το οποίο είναι υπεύθυνο για την τροφοδοσία του κατάλληλου ρεύματος ενεργοποίησης του ηλεκτρονόμου (relay).
- Έναν διαιρέτη τάσης που εφαρμόζεται στη βάση του τρανζίστορ και δίνει την έναυση αγωγιμότητάς του.
- Μια δίοδο για την προστασία υπερφόρτωσης του πηνίου του ηλεκτρονόμου (δίοδος σβέσης)

Με την ενεργοποίηση του ηλεκτρονόμου, δίνοντας την κατάλληλη εντολή από το μικροελεγκτή στην αντίστοιχη θύρα (στο συγκεκριμένο πρόγραμμα έχει οριστεί η 5 ψηφιακή θύρα), δημιουργείται μία τάση στη βάση του τρανζίστορ και το καθιστά αγώγιμο με συνέπεια να δημιουργείται ροή ρεύματος μέσα στο πηνίο του ηλεκτρονόμου. Εν συνεχεία ο ηλεκτρομαγνήτης ελκύει τον οπλισμό των επαφών που είναι συνδεδεμένο το φορτίο, κλείνοντας το κύκλωμα του εναλλασσόμενου ρεύματος. Όταν το τρανζίστορ απενεργοποιείται, η δίοδος κατά μήκος του πηνίου άγει στην αντίθετη κατεύθυνση, έτσι ώστε να προστατεύσει το κύκλωμα από τυχόν αιχμές τάσεων.

**Κύκλωμα χαμηλής ισχύος**

Η δεύτερη πλευρά αποτελεί την τροφοδοσία για τον μικροελεγκτή. Αφορά το κύκλωμα χαμηλής ισχύος στο οποίο εντάσσεται ο μικροελεγκτής και αναλαμβάνει τον έλεγχο της λειτουργίας του ηλεκτρονόμου και την ανάγνωση των δεδομένων του αισθητήρα.

Ο αισθητήρας ACS 712 τροφοδοτεί την αναλογική είσοδο του controller (A0) με τάση 66 mV ανά κάθε Ampere που διαρρέει το εναλλασσόμενο κύκλωμα που είναι συνδεδεμένη η συσκευή κατανάλωσης. Η ψηφιακή έξοδος D5 του μικροελεγκτή είναι αυτή που καθορίζει την ενεργοποίηση/απενεργοποίηση του ηλεκτρονόμου και κατά συνέπεια τον έλεγχο τροφοδοσίας της επιλεγμένης ηλεκτρικής συσκευής (εναλλασσόμενο φορτίο).

Όταν η πειραματική κατασκευή τροφοδοτηθεί με ρεύμα ενεργοποιείται αυτόματα ο ηλεκτρονόμος. Ο αισθητήρας παράγει μια αναλογική τάση, ανάλογη του φορτίου που διαρρέει το εναλλασσόμενο κύκλωμα και την οποία μέσα από το πρόγραμμα του ελεγκτή τη μετατρέπουμε σε ένταση ρεύματος (Ampere).

# Κώδικας

Σε αυτό το σημείο ακολουθεί η ανάλυση του κώδικα του προγράμματος που αναπτύχθηκε και φορτώθηκε στον μικροελεγκτή της πλατφόρμας Arduino. Για κάθε συνάρτηση αντιστοιχεί μια παράγραφος στην οποία πραγματοποιείται η ανάλυσή της.

**Δήλωση των γενικών μεταβλητών**

Αρχικά στο πρόγραμμα γίνεται η δήλωση των σταθερών και των γενικών μεταβλητών. Η χρήση του αισθητήρα ACS712 δεν προϋποθέτει την εγκατάσταση κάποιας βιβλιοθήκης στον μικροελεγκτή [46].

Αναλυτικότερα, δηλώνονται η αναλογική θύρα εισόδου που θα δεχτεί την τάση από τον αισθητήρα και εν συνεχεία ένα σύνολο μεταβλητών που χρησιμοποιούνται για τη διαδικασία υπολογισμού του εναλλασσόμενου ρεύματος. Η τιμή 66 που ορίζεται για τη μεταβλητή mVperAmp είναι ένας συντελεστής που δίνεται από τον κατασκευαστή για τη μετατροπή της εξόδου του αισθητήρα σε μονάδες Ampere, (66mV αναλογούν για κάθε 1A που διαρρέει το κύκλωμα). Η μέγιστη τιμή που μπορεί να πάρει η έξοδος υπολογίζεται ως εξής:

> 66 mV * 30A = 1998mV = 1.98 V

Λόγω του ότι ο αισθητήρας μετρά εναλλασσόμενο ρεύμα, για τη λήψη σωστών μετρήσεων θα πρέπει να εκτιμηθεί η μέση τιμή της τάσης. Για το σκοπό αυτό δηλώνονται και οι μεταβλητές που θα χρησιμοποιηθούν για τον υπολογισμό του RMS, της μέσης τιμής του εναλλασσόμενου ρεύματος [47].

```
const int sensorIn = A0;
int mVperAmp = 66;   // 66 for 1A Module
double VRMS = 0;
double AmpsRMS = 0;
double Voltage = 0;
```

```
double Watts = 0;

void setup()
{
    Serial.begin(9600);
    pinMode(5, OUTPUT);
    digitalWrite(5, HIGH); // Turns ON Relays 1
    delay(1000);
}
```

**Η συνάρτηση setup**

Στη συνάρτηση setup πραγματοποιείται η παραμετροποίηση των στοιχείων του κυκλώματος. Πιο συγκεκριμένα ο μικροελεγκτής τίθεται να ξεκινήσει τη σειριακή του επικοινωνία και ενημερώνεται σχετικά με την ψηφιακή έξοδο που θα χρησιμοποιηθεί. Ο ακροδέκτης 5 προγραμματίζεται να λειτουργεί ως έξοδος μέσω της συνάρτησης PinMode(pin, OUTPUT). Στη συνέχεια ορίζεται η ενεργοποίηση του ηλεκτρονόμου (ρελέ) μέσω της συνάρτησης digitalWrite(5, HIGH). Θέτοντας τη δεύτερη παράμετρο σε HIGH γίνεται η ανάθεση στον ακροδέκτη 5 της τάσης 5 V [48].

```
void loop()
{
    Voltage = getVPP();
    VRMS = (Voltage / 2.0) * 0.707;  //root 2 is 0.707
    AmpsRMS = (VRMS * 1000) / mVperAmp;
    if (AmpsRMS <= 0.02) AmpsRMS = 0;
    Watts = AmpsRMS * 220;
    Serial.print(" Amps RMS = ");
    Serial.print(AmpsRMS);
    Serial.print(" \t Watts = ");
    Serial.println(Watts);
    delay(2500);


}
```

### Η συνάρτηση loop

Στην συνάρτηση loop έχει υλοποιηθεί το κομμάτι του προγράμματος το οποίο εκτελείται επανειλημμένα από τον μικροελεγκτή. Για την καλύτερη κατανόηση και για την αποφυγή της πολυπλοκότητας του κώδικα έχει δημιουργηθεί μία συνάρτηση getVPP().

Στην αρχή της συνάρτησης loop() καλείται η getVPP() για την λήψη της τιμής του εναλλασσόμενου ρεύματος από κορυφή σε κορυφή.

```
float getVPP()
{
    float result;
    int readValue;             // value read from the sensor
    int maxValue = 0;          // store max value here
    int minValue = 1024;       // store min value here

    uint32_t start_time = millis();
    while ((millis() - start_time) < 1000)  // sample for 1 Sec
    {
        readValue = analogRead(sensorIn);
        // see if you have a new maxValue
        if (readValue > maxValue)
        {
            /*record the maximum sensor value*/
            maxValue = readValue;
        }
        if (readValue < minValue)
        {
            /*record the minimum sensor value*/
            minValue = readValue;
        }
    }
```

```
    // Subtract min from max
    result = ((maxValue - minValue) * 5.0) / 1024.0;

    return result;
}
```

Η συνάρτηση αυτή για ένα δευτερόλεπτο (1000ms), παίρνει δειγματοληπτικές τιμές της τάσης του αισθητήρα (μέγιστες και ελάχιστες) προκειμένου να προσδιοριστεί μια μέση τιμή τάσης και να επιτευχθεί μεγαλύτερη ακρίβεια στα ποσοστά της μέτρησης.

Ο αναλογικός μετατροπέας του Arduino έχει ακρίβεια 10 bit και τάση αναφοράς 5 V. Έτσι στην ψηφιακή κλίμακα έχουμε 1024 διατμήσεις των 5V όπου 4.8mV η ελάχιστη διατρητή τιμή που μπορεί να μετρήσει ο μικροελεγκτής.

```
5000mV/1024 = 4.8mV
```

Αυτό σημαίνει ότι η αναλογική τάση εισόδου από τον αισθητήρα μετατρέπεται σε μια αριθμητική τιμή μεταξύ 0-1023. Η ψηφιακή αυτή τιμή (ReadValue) αναπαριστά την αναλογική τιμή εισόδου και δίνει το εύρος τιμών του ADC μετατροπέα.

Η συνάρτηση Millis() επιστρέφει τον αριθμό των χιλιοστών του δευτερολέπτου από τη στιγμή που άρχισε να τρέχει το πρόγραμμα.

Δημιουργούμε μια επανάληψη ανάγνωσης τιμών συνολικής καθυστέρησης ενός δευτερολέπτου, χρόνος αρκετός για τις δειγματοληπτικές τιμές που θα μας φέρουν πιο κοντά στο επιθυμητό αποτέλεσμα.

Η τάση VPP υπολογίζεται από το παρακάτω πηλίκο:

```
result = ((maxValue - minValue) * 5.0)/1024.0;
```

Ο τύπος μας δίνει μια τάση εξόδου η οποία προκύπτει από τη μέτρηση της μέγιστης και ελάχιστης τιμής του μετατροπέα ADC, όπου στη συνέχεια της loop συνάρτησης θα χρησιμοποιηθεί για την εύρεση της μέσης τιμής της τάσης που μετρούμε.

**Συνέχεια της loop**

Η συνάρτηση getVPP() μας επέστρεψε μια τάση peak to peak. Τώρα το μέλημά μας είναι να τη μετατρέψουμε σε μια τάση RMS έτσι ώστε να πάρουμε τη σωστή

αναλογία των μονάδων της έντασης του ρεύματος που διαρρέει το κύκλωμα, λαμβάνοντας υπόψη ότι ο αισθητήρας μας δίνει 66 mV ανά A.

```
VRMS = (Voltage / 2.0) * 0.707;
```

Η μεταβλητή VRMS περιλαμβάνει τη μέση τιμή της τάσης μέτρησης. Υπολογίζοντας την ένταση που διαρρέει το κύκλωμα βρίσκουμε την ισχύ κατανάλωσης. Λαμβάνοντας υπόψιν ότι η τάση του δικτύου είναι 220V μπορούμε να υπολογίζουμε την ισχύ που διαρρέει το κύκλωμά μας.

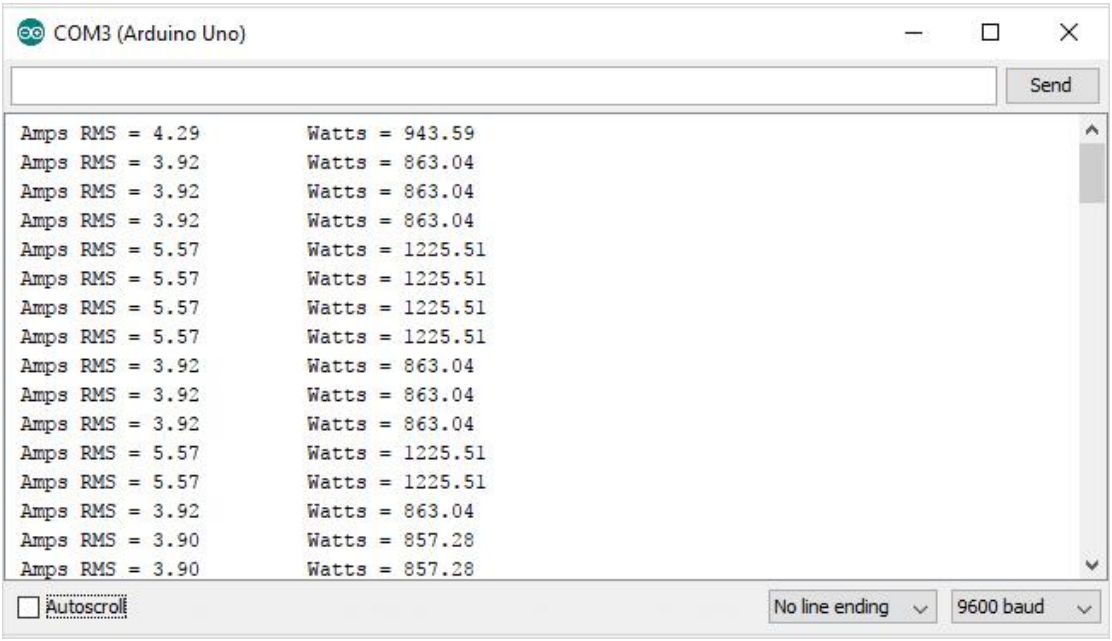
```
AmpsRMS = (VRMS * 1000)/mVperAmp;
```

Στην μεταβλητή AmpsRMS καταχωρείται η ένταση σε μονάδες A, για το λόγο αυτό γίνεται η μετατροπή των mV σε V.

**Διόρθωση σφάλματος**

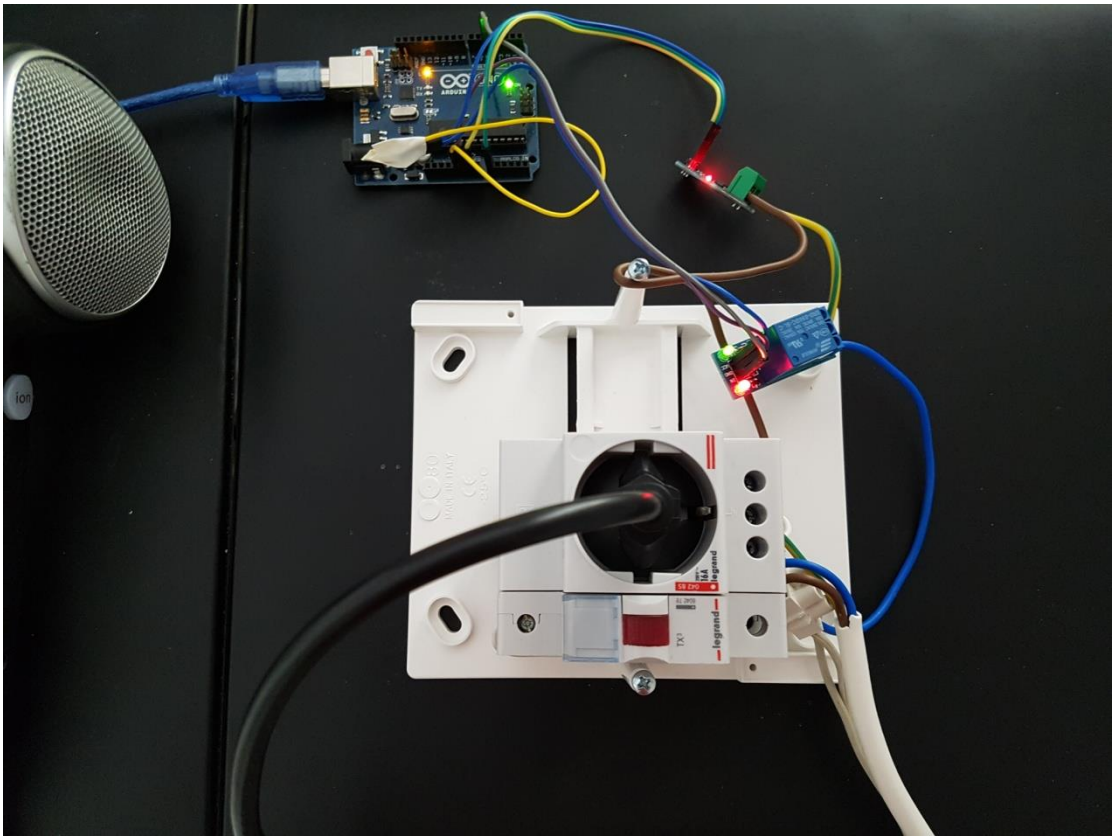
Μια παρατήρηση που σημειώθηκε στις μετρήσεις που διεξήχθησαν, αποτελεί το γεγονός ότι καθώς ο αισθητήρας είναι απενεργοποιημένος, λόγω μαγνητικών πεδίων παράγεται κάποιο ρεύμα το οποίο δεν εξαρτάται από το αν υπάρχει συνδεδεμένη συσκευή. Για να εξαλείψουμε εγγραφές μη πραγματικές, χαμηλής ισχύος, πειραματικά διορθώσαμε αυτό το σφάλμα μηδενίζοντας τιμές ρεύματος κάτω από 0.02A. Οι μικροαλλαγές στις τιμές θεωρούνται αμελητέες και δεν επηρεάζουν την ακρίβεια του μετρητικού μας συστήματος, το οποίο επικεντρώνεται στη μέτρηση κατανάλωσης των οικιακών συσκευών ισχύος.

Στην παρακάτω εικόνα 5.2 λαμβάνουμε κάποιες ενδεικτικές τιμές της εξόδου που μας δίνει στη σειριακή θύρα ο μικροελεγκτής. Συγκεκριμένα για τα εξής δύο μεγέθη: την ένταση ρεύματος και την ισχύ που καταναλώνει η συσκευή (συσκευή στεγνώματος μαλλιών). Αξίζει να αναφερθεί ότι όταν ενεργοποιείται η συσκευή παρατηρούμε ένα ρεύμα έναυσης το οποίο είναι πιο μεγάλο από το ρεύμα κατανάλωσης της και κρατά για ένα μικρό χρονικό διάστημα.




**Εικόνα 5-2: Στιγμιότυπο καταγραφής μετρήσεων**

Στο στιγμιότυπο αυτό καταγραφής γίνονται αντιληπτές κάποιες διακυμάνσεις της κατανάλωσης. Αυτό οφείλεται στα διάφορα σενάρια χρήσης της συσκευής που εφαρμόστηκαν και διαμόρφωσαν τις συνθήκες μέτρησης. Όταν επιχειρήσαμε να αλλάξουμε σκάλα λειτουργίας της συσκευής στεγνώματος μαλλιών και ζητήσαμε μεγαλύτερη θερμότητα (περισσότερο αέρα), η ένταση από 3.9 Α αυξήθηκε στα 5.6 Α περίπου. Το σενάριο επαναφοράς στη χαμηλότερη σκάλα λειτουργίας συνδέεται με τις ενδείξεις 3.92 Α.



**Εικόνα 5-3: 1η πειραματική διάταξη**

Στην εικόνα 5.3 παρατηρούμε τις συνδέσεις της πειραματικής κατασκευής που αφορούν τον αισθητήρα ACS712, το module relay, τον μικροελεγκτή Arduino καθώς και την τροφοδοσία της επιλεγμένης συσκευής κατανάλωσης. Ο μικροελεγκτής είναι συνδεδεμένος σειριακά με τον υπολογιστή.

### 5.2.2 2η Πειραματική Διάταξη

Στη δεύτερη πειραματική διάταξη προχωρήσαμε στην αλλαγή του αισθητήρα μέτρησης έντασης ρεύματος και συγκεκριμένα στη χρήση του SCT013-30 (0-30A).

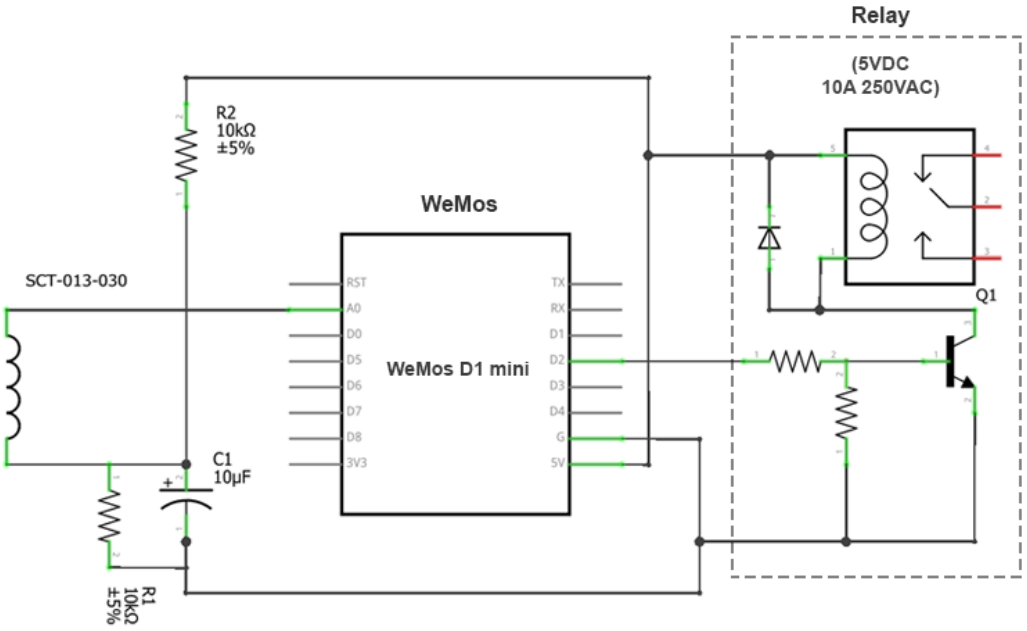
Για να είναι εφικτή η ανάγνωση της τάσης εξόδου του SCT013-30 από τον μικροελεγκτή ATmega 328P του Arduino θα πρέπει να γίνει η μετατροπή της στα όρια των αναλογικών σημάτων που μπορεί να μετρήσει ο συγκεκριμένος μικροελεγκτής. Το εύρος τιμών των αναλογικών σημάτων που μπορεί να διαβάσει ο μικροελεγκτής ATmega 328P είναι από 0-5 V. Επομένως για το λόγο του ότι ο μετατροπέας ADC του ATmega 328P δεν έχει τη δυνατότητα μέτρησης αρνητικών τιμών, προχωρήσαμε στη σύνδεση ενός διαιρέτη τάσης, προσθέτοντας στην τάση εξόδου του μια DC συνιστώσα με τιμή 2.5 V.

Με τον τρόπο αυτό επιτυγχάνεται η προσαρμογή της τάσης σε όλο το αποδεκτό εύρος τιμών, το οποίο κυμαίνεται από 1.5 έως 3.5 V, μία διακύμανση που προκύπτει αν λάβουμε υπόψη την έξοδο του αισθητήρα που είναι από 0-1 V peak to peak (Vpp) για μετρήσεις από 0-30A.

### 5.2.3 3η Πειραματική Διάταξη

Προκειμένου να προσεγγίσουμε τις πραγματικές συνθήκες σε μια πιθανή καταναλωτική χρήση της διάταξής μας, προχωρήσαμε στην εφαρμογή του ασύρματου τρόπου μετάδοσης των μετρήσεων.

Σε αυτό το σενάριο χρησιμοποιήθηκε το wemos mini D1, μικροελεγκτής με δυνατότητες ασύρματης επικοινωνίας και με εύρος τιμών αναλογικών σημάτων από 0-3.3V. Η αναλογική είσοδος του θα πρέπει να είναι μια θετική τάση μεταξύ 0 V και της τάσης αναφοράς ADC του μικροελεγκτή. Αναλογιζόμενοι τα παραπάνω και για λόγους ευκολίας χρήσης κρατήσαμε από το προηγούμενο σενάριο τη σύνδεση με τον ίδιο διαιρέτη τάσης.




**Εικόνα 5-4: Συνδεσμολογία 3ης πειραματικής διάταξης**

**Διάταξη Ελέγχου**

Στην παρούσα διάταξη η εφαρμογή της DC συνιστώσας με τιμή 2.5V, μας δίνει τη δυνατότητα να μετρήσουμε μια διακύμανση +- 0.8V p-p, παίρνοντας ελάχιστη και μέγιστη τιμή 1.7 και 3.3 (+-24A περίπου). Ο περιορισμός της μέγιστης ισχύος που παρουσιάζεται εδώ, συγκριτικά με το προηγούμενο σενάριο, δεν μας επηρεάζει αρνητικά διότι εξακολουθεί να καλύπτεται όλο το φάσμα των μετρήσεων. Για τη διάταξη μας αυτό αντιστοιχεί σε μια μέτρηση γύρω στα 4 KW που είναι απόλυτα επαρκή για την τροφοδότηση των οικιακών συσκευών. Στα πειράματά μας εκτιμώντας ότι το μεγαλύτερο φορτίο που χρησιμοποιήθηκε είναι 10A, δεν ξεπεράσαμε τα 2.2 KW κατανάλωσης.

Αν θέλαμε να μετρήσουμε τα 30 A που μπορεί να μας δώσει ο αισθητήρας SCT0-30 θα έπρεπε να προσαρμόσουμε έναν διαιρέτη τάσης με DC συνιστώσα 2V, παίρνοντας ένα εύρος τιμών από 1-3V που αντιστοιχούν 0-30A. Η διακύμανση της τάσης αυτής ανταποκρίνεται στην τάση εισόδου που μπορεί να μετρήσει ο ADC μετατροπέας του wemos.

**Κώδικας**

Προχωρώντας ένα βήμα παραπάνω στην υλοποίηση του συστήματός μας, ξεκινήσαμε να διερευνούμε το κομμάτι για την αποστολή των μετρήσεων σε ένα web server [49]. Ένα από τα πρώτα πειράματα αποτέλεσε το πρόγραμμα που αναλύεται σε αυτήν την ενότητα, το οποίο αναλαμβάνει να καταγράψει τις τιμές κατανάλωσης που μετρούνται σε ένα αρχείο cvs.

```
# include <ESP8266WiFi.h>
# include <ESP8266WiFiMulti.h>
# include <ESP8266HTTPClient.h>
# include <EmonLib.h>

#define USE_SERIAL Serial
ESP8266WiFiMulti WiFiMulti;

int reading = 0;
int iterations = 10;
```

```
float voltage = 0.0;

EnergyMonitor emon1;
```

Στο άνωθεν κομμάτι κώδικα, αρχικά δηλώνονται οι βιβλιοθήκες των οποίων θα γίνει χρήση (ESP8266, τη βιβλιοθήκη emolib του SCT013-30). Δημιουργούνται δύο μεταβλητές η Serial που ξεκινά τη σειριακή επικοινωνία και η WiFiMulti με την οποία μπορούμε να εισάγουμε (εκ των προτέρων στον κώδικα) πολλά σημεία πρόσβασης - Access Point, που δίνουν τη δυνατότητα να χρησιμοποιήσουμε τη διάταξη σε διαφορετικά δίκτυα. Είναι σημαντικό να τονιστεί ότι στο πλαίσιο των πρώτων πειραμάτων μας η δήλωση των στοιχείων της σύνδεσης του δικτύου γίνεται με προκαθορισμένες τιμές που ανταποκρίνονται στο προσωπικό μας δίκτυο. Στο τελικό στάδιο της παραγωγικής διαδικασίας μια προτεινόμενη λύση είναι η χρήση της βιβλιοθήκης WiFiManager.h.

Οι παράμετροι του SSID και του Password τροφοδοτούνται από ένα Interface που έχει ενσωματωμένο ο μικροελεγκτής και με το οποίο δίνεται η δυνατότητα στο χρήστη να παραμετροποιήσει πολύ εύκολα τα δεδομένα αυτά ορίζοντας τα στοιχεία της σύνδεσης του δικτύου που βρίσκεται ο router του.

```
void setup()
{
    Serial.begin(115200);      // Start Serial
    USE_SERIAL.begin(115200);

    USE_SERIAL.println();
    USE_SERIAL.println();
    USE_SERIAL.println();

    for (uint8_t t = 8; t > 0; t--)
    {
        USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
        USE_SERIAL.flush();
        delay(1000);
    }
```

```
    WiFiMulti.addAP("SSIDRouter", "Password");

    //WiFiMulti.addAP("Router2-SSI", "Router2-password");

    delay(500);

    emon1.current(0, 29); // Current: input pin, calibration. Cur Const= Ratio/BurdenR. 1800/62 = 29.

}
```

Στη setup γίνεται η αρχικοποίηση των διάφορων μεταβλητών. Ο μικροελεγκτής αρχικοποιείται για να μπορέσει να επιτευχθεί η σειριακή επικοινωνία καθώς και η ασύρματη σύνδεση. Η σύνδεση του γίνεται στο Access Point του δικτύου με το προκαθορισμένο SSID και κωδικό. Στη συνέχεια γίνεται η αρχικοποίηση του αισθητήρα ρεύματος για την ανάγνωση των τιμών. Η βιβλιοθήκη του αισθητήρα δέχεται δύο παραμέτρους, την αναλογική είσοδο και η άλλη είναι μια σταθερά που προκύπτει από τις στροφές του πηνίου του αισθητήρα και το ωμικό φορτίο του (στη συγκεκριμένη περίπτωση το 29 – στοιχείο του κατασκευαστή).

```
void loop()

{

    double Irms = emon1.calcIrms(1480); // Calculate Irms only

                                        //double watts= Irms * 220;

    delay(20);

    Serial.println(String(Irms));

    Serial.println(WiFi.localIP());


    HTTPClient http;

    USE_SERIAL.print("[HTTP] begin...\n");

    // configure traged server and url

    http.begin("http://192.168.1.8/wemos/test.php?ipsrc=Test2&crms="       + String(Irms); +"&active=1&state=1"); //HTTP


    USE_SERIAL.print("[HTTP] GET...\n");

    // start connection and send HTTP header

    int httpCode = http.GET();


    // httpCode will be negative on error
```

```
    if (httpCode > 0)
    {
        // HTTP header has been send and Server response header has been handled
        USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

        // file found at server
        if (httpCode == HTTP_CODE_OK)
        {
            String payload = http.getString();
            USE_SERIAL.println(payload);
        }
    }
    else
    {
        USE_SERIAL.printf("[HTTP]    GET...    failed,    error:    %s\n", http.errorToString(httpCode).c_str());
    }
    http.end();
    delay(15000);   // wait 15 seconds
}
```

Στην κύρια συνάρτηση (επανάληψη), διαβάζουμε το ρεύμα του αισθητήρα σε μορφή RMS. Καλείται η συνάρτηση `calcIrms` του αισθητήρα για την αντίστοιχη μέτρηση. Συνακόλουθα, δημιουργούμε ένα http client για να συνδεθούμε σε ένα web server που στον παρόν στάδιο φιλοξενείται σε ένα τοπικό μηχάνημα του δικτύου με τη διεύθυνση `192.168.1.8` , όπως φαίνεται στον κώδικα.

Στη συνέχεια καλείται ένα αρχείο php (test.php) που βρίσκεται στο web server και στο οποίο περνούν μέσω μεταβλητών κάποιες καταστάσεις του μικροελεγκτή και οι μετρήσιμες τιμές του ρεύματος. Το script php αυτό αναλαμβάνει να καταγράψει τις τιμές των παραμέτρων σε ένα αρχείο cvs order. Η διαδικασία επαναλαμβάνεται κάθε 15 sec.

**Test.php**

```
<? php



if(!empty($_GET["crms"])        &&        !empty($_GET["active"])        &&
!empty($_GET["state"]) && !empty($_GET["ipsrc"])){

   $csvData                                                                    =
array($_GET["ipsrc"],$_GET["crms"],$_GET["active"],$_GET["active"],
date("Ymd"),date("H:i:s"));

   $fp = fopen("order.csv","a");

   if($fp)    {

       fputcsv($fp,$csvData); // Write information to the file

       fclose($fp); // Close the file

   }

}

?>
```

### 5.2.4 Συμπεράσματα - Τελική κυκλωματική υλοποίηση

Μετά από έρευνα και ερμηνεύοντας τα αποτελέσματα της μελέτης των προηγούμενων πειραματικών διατάξεων καταλήξαμε σε κάποια συμπεράσματα σχετικά με την επιλογή του μικροελεγκτή και του αισθητήρα ρεύματος.

**Επιλογή Μικροελεγκτή**

Το Arduino uno που χρησιμοποιήσαμε δέχεται shields αρθρωτές επεκτάσεις Wifi, επωμισμένο όμως το βάρος του κόστους και του μεγέθους (αρκετά μεγάλο για την ενσωμάτωσή του σε μικρές κατασκευές). Ο παράγοντας αυτός κατείχε σημαντικό ρόλο στην απόφασή μας να χρησιμοποιήσουμε στο τελικό σύστημά μας τον μικροελεγκτή wemos D1 mini, ο οποίος είναι αρθρωτός, αρκετά μικρός, μπορεί να ενσωματωθεί εύκολα και είναι συμβατός με τον κώδικα του Arduino, ενώ ακόμα διαθέτει δικό του Framework σε Python (Compatible Node mcu, micro python). Επιπλέον ο wemos D1 mini αποτελεί μια οικονομική επιλογή.

**Επιλογή αισθητήρα έντασης ρεύματος**

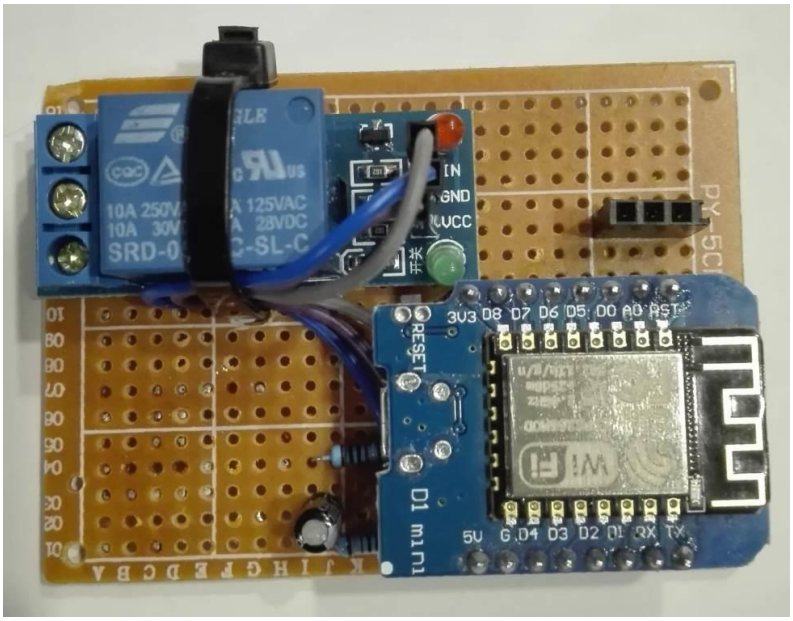
Από τα πειράματα που διεξήχθησαν παρατηρήσαμε ότι η ακρίβεια και των δύο μετρητών έντασης ρεύματος ήταν εφάμιλλη των μετρήσεων του πολυμέτρου (SOAR Multimeter 4040) που χρησιμοποιήθηκε για την σύγκριση των τιμών σε πραγματικό

χρόνο. Έπειτα από συγκριτική μελέτη ωστόσο για τις απαιτήσεις της τελικής διάταξής μας και την πραγματοποίηση των μετρήσεων επιλέχθηκε το μοντέλο μετασχηματιστή ρεύματος SCT013-30 (30A).

Η συγκριτική μελέτη με τα στοιχεία που καθόρισαν την τελική απόφαση της επιλογής του αισθητήρα παρατίθενται παρακάτω:

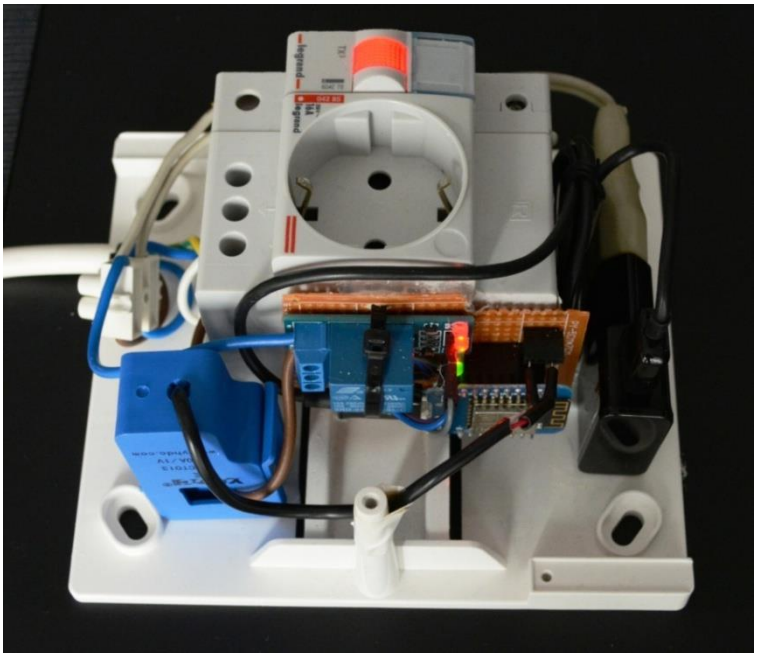
- → Ο αισθητήρας ACS712 είναι μια κατασκευή που βασίζεται σε τυπωμένο κύκλωμα και σε περίπτωση μετρήσεων ρεύματος της τάξης των 30 Α θα μπορούσε να επέλθει βραχυκύκλωμα στους ακροδέκτες της πλακέτας που συνδέεται το εναλλασσόμενο ρεύμα, σενάριο το οποίο θα έθετε σε επικινδυνότητα την κατασκευή μας και προφανώς τη χρήση της. Αυτό ήταν και ένα σημείο μελανό στην επιλογή του αισθητήρα. Ο αισθητήρας SCT013-30 που επιλέξαμε, δεν έρχεται καθόλου σε επαφή με το δίκτυο υψηλής τάσης και έντασης και η μόνη επαφή που έχει με το δίκτυο εναλλασσόμενης τάσης είναι επαγωγική με αποτέλεσμα να εξασφαλίζουμε μεγαλύτερη ασφάλεια για την διάταξη μας. Μπορούμε να λάβουμε πιο εύκολα τα μέτρα προστασίας για το συγκεκριμένο κύκλωμα αποφεύγοντας ηλεκτρικά τόξα και βραχυκυκλώματα λόγω συνδέσεων ηλεκτροφόρων καλωδίων.

- → Ένας άλλος παράγοντας είναι ότι μπορούμε να μετρήσουμε μεγαλύτερες εντάσεις (υπάρχει έκδοση του αισθητήρα για μέτρηση μέχρι και 100Α), άρα και συσκευές μεγαλύτερης κατανάλωσης.




**Εικόνα 5-5: Διάτρητη πλακέτα κατασκευής**

Ένα μέρος της κατασκευής αποτελεί μία διάτρητη πλακέτα στην οποία ενσωματώσαμε τον μικροελεγκτή καθώς και τον module του relay και μια υποδοχή η οποία θα δεχτεί τις ακίδες εξόδου του αισθητήρα ρεύματος SCT013-30. Ο μικροελεγκτής είναι τοποθετημένος πάνω σε μία βάση έτσι ώστε να είναι αποσπώμενος.



**Εικόνα 5-6: 3η πειραματική διάταξη**

Βλέπουμε μια φωτογραφία από τα τελικά στάδια της συνδεσμολογίας όπου φαίνεται η υποδοχή σούκο που θα φιλοξενήσει τη συσκευή κατανάλωσης, ο αισθητήρας ρεύματος SCT013-30 που τον διαπερνά το καλώδιο ισχύος 220V, δεξιά η τροφοδοσία του μικροελεγκτή όπου χρησιμοποιήθηκε ένα τροφοδοτικό Usb 2A.

## 5.3 Διατάξεις Ελέγχου

Πέρα από το κομμάτι της καταγραφής των ενεργειακών δεδομένων, το ενδιαφέρον μας στράφηκε στην επίτευξη της απομακρυσμένης διαχείρισης των συσκευών. Στην ενότητα αυτή παρουσιάζονται σενάρια που σχετίζονται με την υλοποίηση του χρονοπρογραμματισμού.

Δοκιμάσαμε διάφορες εκδοχές ελέγχου της συσκευής. Ακολουθούν τα πειράματα που διεξήχθησαν βασισμένα στην αρχιτεκτονική client – server για να φτάσουμε στην τελική έκδοση βασισμένη στο πρωτόκολλο mqtt όπως θα δούμε στο επόμενο κεφάλαιο.

### 5.3.1 Πρώτη Εκδοχή

Το πρώτο πείραμα που επιχειρήσαμε ήταν να δημιουργήσουμε ένα αρχείο json μέσα στο web server που θα διαβάζει το wemos, και στο οποίο καταγράφεται η ενέργεια που αντιστοιχεί σε ενεργοποίηση ή απενεργοποίηση (on/off) του ηλεκτρονόμου (relay).

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>

const char* ssid = "<SSID>";
const char* password = "********";
const char* host = "192.168.1.8"; // Your domain
const int pin = D2;

String path = "/wemos/relay.json";

void setup()
{
    pinMode(pin, OUTPUT);
```

```
    pinMode(pin, HIGH);
    Serial.begin(115200);
    delay(10);

    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    int wifi_ctr = 0;

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: " + WiFi.localIP());
}

void loop()
{
    Serial.print("connecting to ");
    Serial.println(host);

    WiFiClient client;
    const int httpPort = 80;

    if (!client.connect(host, httpPort))
    {
        Serial.println("connection failed");
        return;
    }
```

```
client.print(String("GET ") + path + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" +
"Connection: keep-alive\r\n\r\n");

delay(500); // wait for server to respond
// read response
String section = "header";

while (client.available())
{
    String line = client.readStringUntil('\r');
    // Serial.print(line);
    // we'll parse the HTML body here

    if (section == "header")
    { // headers..
        Serial.print(".");
        if (line == "\n")
        { // skips the empty space at the beginning
            section = "json";
        }
    }
    else if (section == "json")
    {
        section = "ignore";
        String result = line.substring(1);
        // Parse JSON
        int size = result.length() + 1;
        char json[size];
        result.toCharArray(json, size);
        StaticJsonBuffer < 200 > jsonBuffer;
```

```
            JsonObject & json_parsed = jsonBuffer.parseObject(json);


            if (!json_parsed.success())
            {
                Serial.println("parseObject() failed");
                return;


            }
            // Make the decision to turn off or on the Relay
             // sample of json file: {"relay": "on"}


            if (strcmp(json_parsed["relay"], "on") == 0)
            {
                digitalWrite(pin, HIGH);
                Serial.println("RELAY ON");
            }
            else
            {
                digitalWrite(pin, LOW);
                Serial.println("RELAY OFF");
            }
        }
    }
    Serial.print("closing connection. ");
}
```

Δηλώνονται οι βιβλιοθήκες esp8266 και arduinojson.h (συμβατή με το wemos). Στη συνέχεια κάποιες μεταβλητές που αφορούν το web server, τα στοιχεία του router, τη σύνδεση με το host. Δίνουμε το path του server και το αρχείο json το οποίο θα χρησιμοποιήσει. Ενεργοποιούμε την έξοδο D2. Κάνουμε parsing το json για να λάβουμε την τιμή από τη μεταβλητή “relay” και αναλόγως ενεργοποιούμε/απενεργοποιούμε το module relay.

Εντός της void setup γίνεται η αρχικοποίηση, εκτελείται η WiFi.begin() με παραμέτρους το SSID και το αντίστοιχο passoword για την εκκίνηση των παραμέτρων του δικτύου. Εκτελείται μια επανάληψη μέχρι η σύνδεσή μας να επιτευχθεί.

Εντός της void loop γίνεται η σύνδεση με το server και διαβάζουμε το json κάνοντας Parse το html που μας επιστρέφει το result.

### 5.3.2 Δεύτερη Εκδοχή

Στο προηγούμενο παράδειγμα η υλοποίηση βασιζόταν στο σενάριο της λειτουργίας του wemos σε client. Κάθε 15 sec γινόταν μια κλήση στο server για να θέσει σε ενεργοποίηση / απενεργοποίηση τη συσκευή. Το γεγονός αυτό θέτει δύο περιορισμούς:

- Πρώτον, η επανάληψη κλήσεων θα έπρεπε να γίνεται σε ένα πολύ στενό χρονικό περιθώριο (κάποιων δευτερολέπτων) για να έχουμε μια σχετική ακρίβεια χρόνου για την υλοποίηση του χρονοπρογραμματισμού.
- Δεύτερον, οι συνεχόμενες κλήσεις του client προς το server θα επέφεραν μεγαλύτερη κίνηση δεδομένων και συνεπώς θα απαιτούνταν μεγαλύτερο δικτυακό εύρος (Bandwidth). Σε πραγματικές συνθήκες στην περίπτωση της ταυτόχρονης διαχείρισης μιας πληθώρας μικροελεγκτών για τον έλεγχο των συσκευών θα δημιουργούνταν ακόμη περισσότερος φόρτος και η ισχύς του server θα έπρεπε να ανταποκρίνεται στις χιλιάδες κλήσεις των clients.

Σε αυτή την εκδοχή το wemos έχει το ρόλο του διακομιστή (λειτουργία web server). Ο κώδικας της σύνδεσης με το WiFi Access Point είναι ίδιος.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>


ESP8266WiFiMulti wifiMulti;          // Create an instance of the
ESP8266WiFiMulti class, called 'wifiMulti'
```

```
ESP8266WebServer server(80);     // Create a webserver object that listens
for HTTP request on port 80


const int relay = D2;


void handleRoot();              // function prototypes for HTTP handlers

void handleInitiate();

void handleNotFound();

void handleRELAY();


void setup(void)

{

    Serial.begin(115200);         // Start the Serial communication to send
messages to the computer

    delay(10);

    Serial.println('\n');


    pinMode(relay, OUTPUT);


    wifiMulti.addAP("SSID", "Password");    // add Wi-Fi networks you want
to connect to

    Serial.println("Connecting ...");

    int i = 0;

    while (wifiMulti.run() != WL_CONNECTED)

    {

        delay(250);

        Serial.print('.');

    }

    Serial.println('\n');

    Serial.print("Connected to ");

    Serial.println(WiFi.SSID());

    Serial.print("IP address:\t");

    Serial.println(WiFi.localIP());
```

```
    server.on("/", HTTP_GET, handleRoot);          // Call the 'handleRoot'
function when a client requests URI "/"

    server.on("/Initiate",   HTTP_POST,   handleInitiate);   //   Call   the
'handleInitiate' function when a POST request is made to URI "/Initiate"

    server.on("/RELAY", HTTP_POST, handleRELAY);  // Call the 'handleRELAY'
function when a POST request is made to URI "/RELAY"

    server.onNotFound(handleNotFound);

    server.begin();

    Serial.println("HTTP server started");

}



void loop(void)

{

    server.handleClient();                      // Listen for HTTP requests
from clients

}



void handleRoot()

{                           // When URI / is requested, send a web page with
a button to toggle the RELAY

    server.send(200,      "text/html",      "<form      action=\"/Initiate\"
method=\"POST\">Device    sn:</br><input    type=\"text\"    name=\"cnt_sn\"
placeholder=\"cnt_sn\"></br>Status:</br><input                type=\"text\"
name=\"cnt_set\"     placeholder=\"cnt_set\"></br><input     type=\"submit\"
value=\"submit\"></form><p>Try   sn   'CZX12345'   and   cnt_set   'on/off'
...</p>");

}

void handleInitiate()

{                           // If a POST request is made to URI /Initiate

    if (!server.hasArg("cnt_sn") || !server.hasArg("cnt_set")

          || server.arg("cnt_sn") == NULL || server.arg("cnt_set") == NULL)

    { // If the POST request doesn't have sn and status data

        server.send(400, "text/plain", "400: Invalid Request");          //
The request is invalid, so send HTTP status 400

        return;

    }
```

```
    if (server.arg("cnt_sn") == "CZX12345")
    {
        if (server.arg("cnt_set") == "on")
        {
            digitalWrite(relay, HIGH);
            server.send(200, "text/plain", "cnt_active_on");
        }
        else
        {
            digitalWrite(relay, LOW);
            server.send(200, "text/plain", "cnt_active_off");
        }
    }
    else
    {
        server.send(401, "text/plain", "401: error_sn");
    }
}
void handleNotFound()
{
    server.send(404, "text/plain", "404: Not found"); // Send HTTP status
404 (Not Found) when there's no handler for the URI in the request
}


void handleRELAY()
{                           // If a POST request is made to URI /RELAY
    digitalWrite(relay, !digitalRead(relay));       // Change the state of
the RELAY
    server.sendHeader("Location", "/");          // Add a header to respond
with a new location for the browser to go to the home page again
}
```

Με τη εισαγωγή της βιβλιοθήκης `ESP8266WebServer.h`, το ESP8266 μπορεί να επιτελέσει τις λειτουργίες ενός διακομιστή ιστού. Αυτό σημαίνει ότι το wemos εκτός

του ότι θα μπορεί να στέλνει μετρήσεις σε έναν διακομιστή στο ρόλο του client, επιπλέον ως server θα έχει τη δυνατότητα να δέχεται αιτήματα για ενεργοποίηση / απενεργοποίηση συσκευών.

Για την επικοινωνία με το server δημιουργούνται τρία routes:

- `"/"` (handleRoot()): ο server θα πρέπει να ενημερωθεί για τη μέθοδο που θα αναλάβει να εξυπηρετεί τις κλήσεις στο root. Ως αρχική σελίδα εμφανίζεται μία φόρμα με δύο πεδία, ο σειριακός αριθμός της συσκευής και η κατάσταση του relay που θέλουμε να θέσουμε
- `"/Initiate"` (handleIntitiate()): γίνεται ο έλεγχος των παραμέτρων και ανάλογα με τις τιμές ενεργοποιεί ή απενεργοποιεί το relay.
- `"/RELAY"` (handleRELAY()): αλλάζει την κατάσταση του relay. Τοποθετώντας το path στο url αντιστρέφει την κατάσταση λειτουργίας του relay (αν το relay είναι on γίνεται off και αντίστροφα)

### 5.3.3 Σύνοψη

Μελετώντας τις παραπάνω εκδοχές σκεφτήκαμε αρχικά ότι μια ενδεδειγμένη λύση για το τελικό σύστημα, με την προσθήκη της υλοποίησης του χρονοπρογραμματισμού, είναι ο μικροελεγκτής (wemos) να λειτουργεί και σε κατάσταση client και σε server. Client για την αποστολή των τιμών των μετρήσεων που σημειώνονται, και server για να μπορεί το σύστημα να διεκπεραιώνει τις ενέργειες χρονοπρογραμματισμού που θα δέχεται από το χρήστη.

Υπάρχουν όμως κάποιοι παράμετροι που δε λάβαμε υπόψη στα πειράματά μας και σχετίζονται με πιθανές δυσκολίες που μπορεί να συναντήσουμε σε πραγματικές συνθήκες. Τα πειράματα μας εδώ διεξήχθησαν στο τοπικό δίκτυο. Στην περίπτωση που η πλατφόρμα μας λειτουργούσε διαδικτυακά θα έπρεπε να υλοποιούνται κατάλληλες ρυθμίσεις για τον δρομολογητή του κάθε τοπικού δικτύου, έτσι ώστε να διοχετεύει τις εντολές της πλατφόρμας στον αντίστοιχο μικροελεγκτή. Ένα πλήθος παραμετροποιήσεων που δυσκολεύει την εγκατάσταση του συστήματος (απαίτηση για forward port) ενώ επιπλέον απαιτεί την περαιτέρω διερεύνηση σε ζητήματα ασφαλείας [50].

Η επικρατέστερη λύση που προέκυψε για την επίτευξη των παραπάνω λειτουργιών του συστήματός μας, είναι μια αρχιτεκτονική βασισμένη στο πρωτόκολλο MQTT

όπως θα δούμε στο επόμενο κεφάλαιο. Με την χρήση του πρωτοκόλλου MQTT έχουμε τη δυνατότητα επιπλέον να εφαρμόσουμε και την ποιότητα υπηρεσιών QoS (ανάλυση του QoS έγινε στο 2$^{ο}$ κεφάλαιο) προκειμένου να εξασφαλίσουμε την εγγύτητα παράδοσης των μηνυμάτων ακόμη και σε περιπτώσεις ξαφνικής διακοπής της λειτουργίας του microcontroller.

# 6 Ανάλυση Τελικού Συστήματος

Καταλήγοντας, δημιουργήθηκε ένα ενοποιημένο σύστημα που αποτελεί τη διεπαφή για τον έλεγχο και την παρακολούθηση της πλατφόρμας αυτοματισμού στα πλαίσια αυτής της μελέτης. Πραγματοποιήθηκαν μετρήσεις σε οικιακό περιβάλλον ώστε να διαπιστωθεί η ακρίβεια της μετρητικής διάταξης, του συμπληρωματικού εξοπλισμού και κυρίως της επικοινωνίας του τελικού συστήματος.

## 6.1 Επικοινωνία Συστήματος

Η επικοινωνία του τελικού συστήματος βασίζεται στο πρωτόκολλο MQTT και στη αρχιτεκτονική του publish - subscribe. Όπως αναλύθηκε στο 2$^{ο}$ κεφάλαιο υπεύθυνος για τη διανομή των μηνυμάτων στους ενδιαφερόμενους συνδρομητές (clients) είναι μια ενδιάμεση οντότητα, ένας ενδιάμεσος διακομιστής, ο Broker. Η εγκατάσταση του διακομιστή είναι αναγκαία για να γίνει η χρήση του πρωτοκόλλου. Στην περίπτωσή μας επιλέχθηκε ο Mosquitto Broker.

### 6.1.1 Από τον μικροελεγκτή στον Broker και αντίστροφα

Όλες τις λειτουργίες του συστήματός μας όπως τις μετρήσεις που πήραμε από τον μετασχηματιστή ρεύματος θα πρέπει να τις μεταφέρουμε στον MQTT Broker, ο οποίος είναι υπεύθυνος για να κρατήσει τις τιμές σε ένα topic και να ενημερώνει τα πεδία αυτά μέσω των publish/subscribe.

Αρχικά θα πρέπει να ενσωματώσουμε τη βιβλιοθήκη PubSubClient.h στον wemos και να δηλώσουμε κάποιες βασικές παραμέτρους ενεργοποίησής της. Στη συνέχεια δημιουργούμε μια συνάρτηση όπου συνδέεται το σύστημά μας με τον broker. Σε πρώτο στάδιο δημιουργούμε τον mqtt Client, του δηλώνουμε τον broker με τον οποίο θα συνδεθεί και την πόρτα. Ακολούθως θα πρέπει να δηλώσουμε την εγγραφή μας, η οποία παίρνει τρία ορίσματα, το πρώτο είναι ο client που δημιουργήσαμε, το δεύτερο το μονοπάτι όπου θα κάνουμε εγγραφή και το τρίτο το QoS.

Η συνάρτηση void setup περιέχει την αρχικοποίηση των τιμών και την εγκαθίδρυση σύνδεσης για τη συσκευής μας στο δίκτυο.

Στη συνάρτηση void loop() υλοποιείται το κομμάτι κώδικα της κλήσης των συναρτήσεων που είναι υπεύθυνες για τη σύνδεση στο Broker, για την αποστολή των τιμών στον Broker καθώς και την ταυτοποίηση των εγγραφών.

Αμέσως μετά δημιουργούμε μια συνάρτηση τη void callback() η οποία είναι υπεύθυνη για τη διαδικασία του προγραμματισμού ενεργοποίησης/απενεργοποίησης των συσκευών, τη συνάρτηση calcWatt για την ενημέρωση του Broker με τις εγγραφές κατανάλωσης του ρεύματος των συσκευών καθώς και τη συνάρτηση checkbound η οποία ελέγχει τη διακύμανση μεταξύ δύο διαδοχικών μετρήσεων. Σε μικρές διακυμάνσεις ισχύος (που καθορίζεται από τη μεταβλητή dif_watts) δεν αποστέλλονται οι τιμές στον Broker αποφεύγοντας ένα σενάριο επιβάρυνσης του δικτύου με επιπλέον φόρτο.

Τέλος χρησιμοποιούμε ακόμη μία συνάρτηση τη void MQTT_reconnect() η οποία είναι υπεύθυνη για να κρατάει τη σύνδεση με τον Broker ανοιχτή, και καλείται εάν χαθεί η σύνδεση με τον Broker έως ότου εγκατασταθεί εκ νέου.

Ενδεικτικά δίνεται ο κώδικας της callback και της getWatts()

```
void callback(char* topic, byte* payload, unsigned int length)
{
    Serial.print("Command from MQTT broker is : [");
    Serial.print(topic);
    Serial.print("], ");
    payload[length] = '\0';
    String message = (char*)payload;
    if (message == "0")
    {
        digitalWrite(CTRL_PIN, LOW);
        ACT_CTRL_PIN = false;
        Serial.println(" Turn Off CONTROLLER! ");
    }
    if (message == "1")
    {
        digitalWrite(CTRL_PIN, HIGH);
        ACT_CTRL_PIN = true;
        Serial.println(" Turn On CONTROLLER! ");
    }
```

```
    Serial.println();
}
```

Μεταβλητές που σχετίζονται με τη μέθοδο getWatts()

```
float watts = 0.0;
float dif_watts = 4.0;
unsigned long previousMillis = 0;
const long delayMillis = 15000;  // 15sec
char data[80];
```

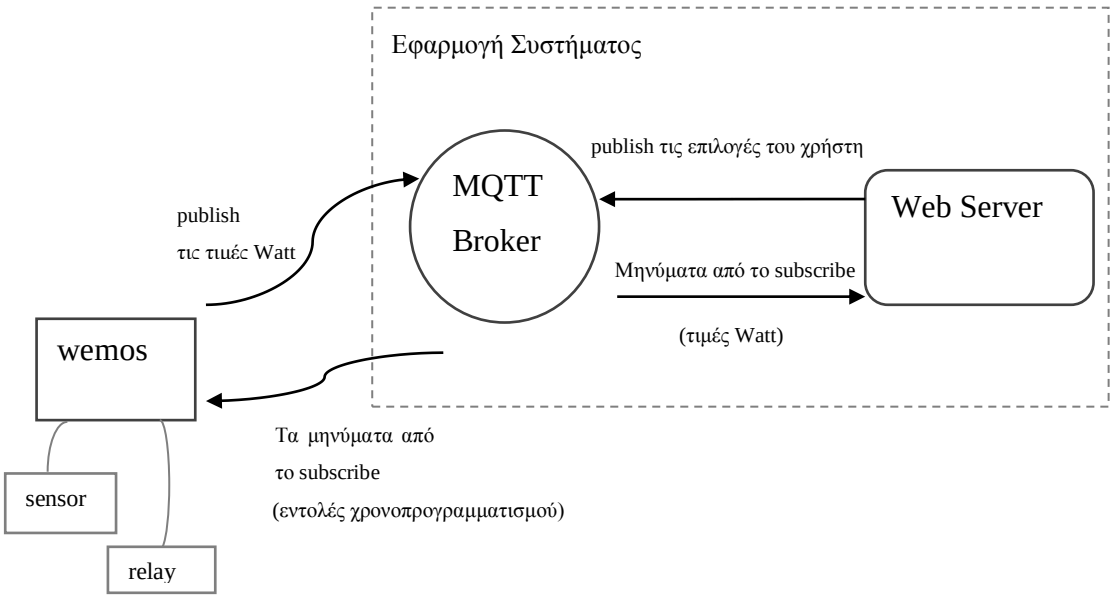
```
void calcWatt ()
{
    // float or double watts; client.publish(topic, String(watts).c_str(), true);
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= delayMillis)
    {
        previousMillis = currentMillis;

        float Irms = emon1.calcIrms(1480); // Calculate Irms only
        float newWatts = Irms * 220;

        if (checkBound(newWatts, watts, dif_watts))
        {
            watts = newWatts;
            snprintf(data, 80, "%ld", (int)watts);
            Serial.print("New Power:");
            Serial.println(String((int)watts).c_str());
            client.publish(pubTopic, data, true);
        }
    }
}
```

```
bool checkBound(float newValue, float prevValue, float maxDiff)
{
    return !isnan(newValue) && (newValue < prevValue - maxDiff || newValue
> prevValue + maxDiff);
}
```

## 6.2 Αρχιτεκτονική

Η ασύρματη μορφή επικοινωνίας μεταξύ των ρευματοδοτών (ελεγκτών) και της εφαρμογής του συστήματος σε συνδυασμό με την εφαρμογή του πρωτοκόλλου MQTT αποτελούν τα θεμελιώδη στοιχεία της αρχιτεκτονικής της τελικής πλατφόρμας.

Στο σκαρίφημα που ακολουθεί, απεικονίζεται ο υπολογιστής εξυπηρετητής του συστήματος, στον οποίο εκτελείται ο web server για την λειτουργία της διαδικτυακής εφαρμογής. Σε αυτόν επίσης, εκτελείται και ο MQTT Broker.




**Εικόνα 6-1: Αρχιτεκτονική τελικού συστήματος**

Το πρωτόκολλο ορίζει πώς οι Clients (του μικροελεγκτή και του web server) πρέπει να κάνουν εγγραφή σε ένα θέμα (topic) του MQTT Broker ώστε να υπάρχει αμφίδρομη ανταλλαγή δεδομένων για την καταγραφή των μετρήσεων καθώς επίσης

και για τη διαχείριση των μικροελεγκτών. Το όνομα αυτού του θέματος θα πρέπει να συνδυάζει και τον σειριακό αριθμό του εκάστοτε οικιακού ελεγκτή για την μοναδικότητα της επικοινωνίας του με τον Broker.

Ο web server κάνει εγγραφή στο μονοπάτι που έχουμε προκαθορίσει για τις τιμές του αισθητήρα ρεύματος από τον client (μικροελεκγτής wemos). Επιπλέον δημοσιεύει (publish) τις επιλογές του χρήστη για τον έλεγχο του ρευματοδότη (σενάριο χρονοπρογραμματισμού).

Από την αριστερή πλευρά βλέπουμε ότι ο μικροελεκτής wemos κάνει subscribe για να πάρει τις εντολές χρονοπρογραμματισμού του χρήστη καθώς επίσης εκτελεί και τη διαδικασία δημοσίευσης (publish) για το σενάριο αποστολής τιμών των μετρήσεων κατανάλωσης ρεύματος.

## 6.3 Οργάνωση Βάσης Δεδομένων

### 6.3.1 Entity Framework και Code First

Η βάση δεδομένων κάθε συστήματος αποτελεί τον πυρήνα του. Στην παρούσα εφαρμογή χρησιμοποιήθηκε το Entity Framework της Microsoft για την οργάνωση, υλοποίηση και πρόσβαση σε δεδομένα της βάσης[7]. Επίσης ακολουθήθηκε η μεθοδολογία της Code First προσέγγισης όπου δε χρειάστηκε να δημιουργηθεί μια βάση δεδομένων πάνω στην οποία θα δεθεί η εφαρμογή, αντιθέτως η βάση δημιουργήθηκε με τη βοήθεια της παραπάνω προσέγγισης από τον κώδικα [51].

#### *6.3.1.1 Code First Προσέγγιση*

**Migrations**

Όπως έχει ήδη προαναφερθεί χρησιμοποιώντας την code first τεχνική κάθε φορά που αλλάζουμε τις βασικές κλάσεις μας που αποτελούν την εφαρμογή μας, αλλάζει και το Schema ή αλλιώς η δομή της βάσης μας. Ο φάκελος migrations περιέχει όλες τις διαφορετικές καταστάσεις από τις οποίες θα περάσει η βάση μας.

Η εφαρμογή χρησιμοποιεί την προκαθορισμένη βιβλιοθήκη της Microsoft για την πιστοποίηση χρηστών και άλλων διάφορων λειτουργιών σχετικά με τους χρήστες την

[7] Στα πλαίσια των πειραμάτων χρησιμοποιήθηκε η SQL Server localDB. Σε πραγματικές συνθήκες λειτουργίας θα πρέπει να επιλεχθεί μια βάση δεδομένων που υποστηρίζεται από το EF, όπως η SQL Server, MySQL, MariaDB, postgreSQL, Oracle.

ASP.NET identity. Με αυτήν τη βιβλιοθήκη υπάρχουν έτοιμοι controllers (ελεγκτές), views (όψεις) αλλά και models (οντότητες) που υλοποιούν λειτουργίες σχετικά με τους χρήστες. Στην δημιουργία του 1ο Migration αν και δεν έχουν οριστεί άλλες κλάσεις δημιουργούνται πίνακες στη βάση δεδομένων όπως :

- AspNetRoles
- AspNetUserClaims
- AspNetUserLogins
- AspNetUserRoles
- AspNetUsers

Η εντολή που πληκτρολογούμε στο package manager console για την πρόσθεση του Migration είναι:

> **add-migration ExampleMigrationName**

Το αποτέλεσμα της εντολής είναι η δημιουργία μιας csharp κλάσης που ορίζει, δημιουργεί τους παραπάνω πίνακες.

Κάθε migration αποτελείται από μια Up μέθοδο και μια Down. Στην ουσία αυτές οι δύο μέθοδοι περιέχουν ακριβώς τις αντίθετες εντολές. Στην Up μέθοδο δημιουργούνται οι SQL πίνακες που δηλώνονται αυτόματα όπως προαναφέραμε μέσω του Identity ενώ στην down διαγράφονται.

**Ενημέρωση της Βάσης**

Το επόμενο βήμα μετά από το κάθε migration είναι η ενημέρωση της βάσης και η πραγματική εκτέλεση των εντολών που δημιουργήθηκαν από το migration. Η εντολή έχει ως εξής:

> **Update-database**

Με την εκτέλεση αυτής της εντολής δημιουργείται ένα αρχείο .mdf που στην ουσία αποτελεί την βάση και ένας φάκελος _MigrationsHistory ο οποίος περιέχει το migration που έτρεξε τελευταίο, δηλαδή την παρούσα κατάσταση της βάσης κάθε στιγμή.

Αφού προηγηθεί η σύνταξη όλων των κλάσεων-οντοτήτων που σχετίζονται με την εφαρμογή, θα εκτελεστεί η δημιουργία του τελικού migration και συνεπώς θα

ενημερωθεί η βάση. Η τελική έκδοση της βάσης είναι οργανωμένη σε πίνακες που σχετίζονται με την καταγραφή των τιμών των μετρήσεων κατανάλωσης, με το κομμάτι του χρονοπρογραμματισμού και οι υπόλοιποι με τη διαχείριση χρηστών (όπως αναφέρθηκε παραπάνω). Αυτοί δεν είναι ανεξάρτητοι, αντιθέτως συνδέονται με σχέσεις που εκφράζονται στα σχεσιακά μοντέλα.

#### *6.3.1.2 Οντότητες Διαχείρισης Των Μετρήσεων Κατανάλωσης*

Σε αυτήν την υποενότητα παρατίθενται κάποιες βασικές οντότητες που υλοποιούν το κομμάτι της ενεργειακής παρακολούθησης.

**Domain Model Device.cs**

Η κλάση Device.cs αντιπροσωπεύει τον πίνακα Device που διατηρούνται στοιχεία σχετικά με βασικές πληροφορίες της συσκευής που θα συνδεθεί στον οικιακό ελεγκτή. Χαρακτηριστικά πεδία του πίνακα αυτού αποτελούν τα: DeviceID, Brand, Type, Status, IoTPCID. Παρακάτω δίνονται αναλυτικές πληροφορίες για κάθε ένα από τα αυτά.

_***DeviceID:*** Δηλώνει το ID της κάθε συσκευής. Επίσης, έχει οριστεί να αυξάνεται από το σύστημα κατά μία μονάδα κάθε φορά που εισάγεται μια επιπλέον συσκευή.

_ ***Brand :*** Αποθηκεύει τη εταιρεία κατασκευής της συσκευής που εισάγει ο χρήστης

_ ***Status :*** Δηλώνει την κατάσταση (ενεργή/ανενεργή)

_ ***Type:*** Αποθηκεύει τον τύπο της συσκευής

_ ***IoTPCID*** : Είναι το ξένο κλειδί που δείχνει το πεδίο ID του πίνακα IoTPC, χρησιμοποιείται για τη σύνδεση της συσκευής με έναν από τους οικιακούς ελεγκτές.

**Domain Model IoTPC.cs (IoTPowerController)**

Κάθε συσκευή συνδέεται σε ένα οικιακό ελεγκτή (κατασκευή-πρίζα). Κάθε γραμμή του πίνακα IoTPC αντιπροσωπεύει την οντότητα ενός οικιακού-ελεγκτή.

_***UserID:*** Δηλώνει το ID του χρήστη

_ ***ID:*** Δηλώνει το ID του κάθε ελεγκτή. Επίσης, έχει οριστεί να αυξάνεται

από το σύστημα κατά μία μονάδα κάθε φορά που εισάγεται ένας επιπλέον ελεγκτής.

_***SerialNumber:*** Μοναδικός σειριακός αριθμός, χρησιμοποιείται στην ταυτοποίηση του ελεγκτή για την λήψη των τιμών των μετρήσεων.

_ ***Description :*** Αποθηκεύει την περιγραφή που εισάγει ο χρήστης (ενδεικτικά: πρίζα-τραπεζαρίας)

_ ***EnrollmentDate :*** Αποθηκεύει το στιγμιότυπο της εγγραφής του ελεγκτή από το χρήστη στην πλατφόρμα.

**Domain Model ConsumptionDetails.cs**

Η οντότητα ConsumptionDetails αντιπροσωπεύει τον πίνακα της εφαρμογής στον οποίο γίνεται αποθήκευση και ανάκτηση των τιμών των μετρήσεων κατανάλωσης ισχύος των συσκευών. Περιέχει πληροφορίες σχετικά με τις μετρήσεις της κάθε συσκευής.

***_ConsumptionDetailsID:*** Δηλώνει το ID της κάθε εγγραφής μέτρησης. Έχει οριστεί να αυξάνεται από το σύστημα κατά μία μονάδα κάθε φορά που γίνεται νέα εισαγωγή εγγραφής.

_ ***Psum :*** Αποθηκεύει την τιμή της μέτρησης που γίνεται ανά 15 δευτερόλεπτα

_ ***CreateTimestamp :*** Αποθηκεύει το στιγμιότυπο της δημιουργίας της μέτρησης

_ ***IoTPCID*** : Είναι το ξένο κλειδί που δείχνει το πεδίο ID του πίνακα IoTPC

***_DeviceID:*** Είναι το ξένο κλειδί που δηλώνει το ID της συνδεδεμένης συσκευής για την οποία γίνεται η μέτρηση.

Για την κατανόηση του τρόπου δημιουργίας των πινάκων, αναλύεται ενδεικτικά ο πηγαίος κώδικας του domain model (κλάσης) ConsumptionDetails.

```
public class ConsumptionDetails
{
    public int ConsumptionDetailsID { get; set; }


    [Required]
    [StringLength(50)]
    [Display(Name = "Device Name")]
    public int DeviceID { get; set; }


    public int IoTPCID { get; set; }
```

```
        [Display(Name = "Power Consumption Details")]

        public double Psum { get; set; }



        [Display(Name = "Consumption Timestamp")]

        [DataType(DataType.DateTime)]

        [DisplayFormat(DataFormatString     =     "{0:dd/MM/yyyy     H:mm}",
ApplyFormatInEditMode = true)]

        public DateTime CreateTimestamp { get; set; }



        //references IoTCP and there is no column that it can figure out
(IoTPCID), so it creates one to try to make the connection between the
tables.

        //public IoTPC IoTPC { get; set; }

        public Device Device { get; set; }



    }
```

Το property DeviceID και IoTPCID υλοποιούν το foreign key constraint όπου εφόσον θα δημιουργηθεί το migration θα δημιουργηθούν και οι στήλες στο συγκεκριμένο πίνακα με τα ξένα κλειδιά που αντιστοιχούν στους πίνακες Device και IoTPC. Επιπλέον παρατηρούμε κάποιες μεταβλητές που υπακούουν σε συμβάσεις που δίνει το Entity Framework για να υποστηρίξει συσχετίσεις μεταξύ πινάκων.

Δίνοντας μια αναλυτική σκοπιά, είναι εμφανές ότι η κλάση είναι μια απλή C # κλάση, ενώ το μόνο αξιοσημείωτο είναι ότι τα πεδία της έχουν πλαισιωθεί με ιδιότητες που ορίζονται μέσα σε τετράγωνες αγκύλες ( [ ] ). Κάποιες από αυτές τις ιδιότητες είναι οδηγίες για το πώς θα εμφανίζονται τα πεδία αυτά και κάποιες άλλες είναι ιδιότητες για το πώς θα αποθηκεύονται στην βάση δεδομένων.

Για την καλύτερη κατανόηση του τι ακριβώς συμβαίνει, ως παράδειγμα για το Domain αυτό Model θα αναλυθούν οι παρακάτω ιδιότητες (properties):

```
1. [Required]
2. [StringLength(50)]
3. [Display(Name = " Consumption Timestamp ")]
4. [DataType(DataType.DateTime)]
5. [DisplayFormat(DataFormatString      =      "{0:dd-MM-yyyy      H:mm}",
   ApplyFormatInEditMode = true)]
```

Στην δήλωση των ιδιοτήτων 1, 2 παρατηρούμε συγκεκριμένες δεσμευμένες λέξεις όπως [REQUIRED], [StringLength(50)]. Η συγκεκριμένη τεχνική ονομάζεται data annotations και δίνει έλεγχο εγκυρότητας στα δεδομένα που πρόκειται να περάσουν στα συγκεκριμένα αντικείμενα καθώς και στα στοιχεία που τελικώς θα αποθηκευτούν στους αντίστοιχους πίνακες στη βάση. Συγκεκριμένα το Required στην ουσία θα μεταφραστεί ως NOT NULL στη βάση και το StringLength (50) ως ένα varchar με μέγιστο αριθμό χαρακτήρων ίσο με 50.

Η ιδιότητα 3 είναι μια οδηγία για την εμφάνιση (View) σύμφωνα με την οποία, σε κάθε περίπτωση που θα χρειαστεί θα εμφανίζεται το όνομα που περιέχεται στο Display και όχι το όνομα του πεδίου. Παρόμοια περίπτωση είναι και η ιδιότητα 5, θα εμφανιστεί η ημερομηνία με το format που επιλέχθηκε στην ιδιότητα.

Η 4η ιδιότητα εδώ αφορά τον τύπο (DataType), είναι μια οδηγία προς τη βάση δεδομένων για το είδος του τύπου του πεδίου CreateTimestamp.

**Δημιουργία πινάκων από τα Domain Models**

Το Entity Framework παρέχει ένα set από κλάσεις που αντιπροσωπεύουν τη σύνδεση μας με τη βάση και τους πίνακές της.

DbContext: Αυτή η κλάση παρέχει όλες τις μεθόδους που μας δίνουν τη δυνατότητα να συνδεόμαστε με τη βάση.

DbSet: Αυτή η κλάση στην ουσία αντιπροσωπεύει τους πίνακές μας.

Παρακάτω φαίνεται η κλάση ApplicationDbContext η οποία υλοποιεί τα παραπάνω.

```
    public class PowerConsumptionContext : DbContext
    {
        public
PowerConsumptionContext(DbContextOptions<PowerConsumptionContext>   options)
: base(options)
        {
        }
        public DbSet<Device> Devices { get; set; }
        public DbSet<IoTPC> IoTPCs { get; set; }
        public DbSet<ConsumptionDetails> ConsumptionsDetails { get; set; }
```

```
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Device>().ToTable("Device");
            modelBuilder.Entity<IoTPC>().ToTable("IoTPC");

modelBuilder.Entity<ConsumptionDetails>().ToTable("ConsumptionDetails");
        }
    }
```

Στην δήλωση των DbSet δίνουμε σαν παράμετρο τα domain models. Αφού προσθέσουμε το νέο migration δημιουργούνται οι καινούριοι πίνακες.

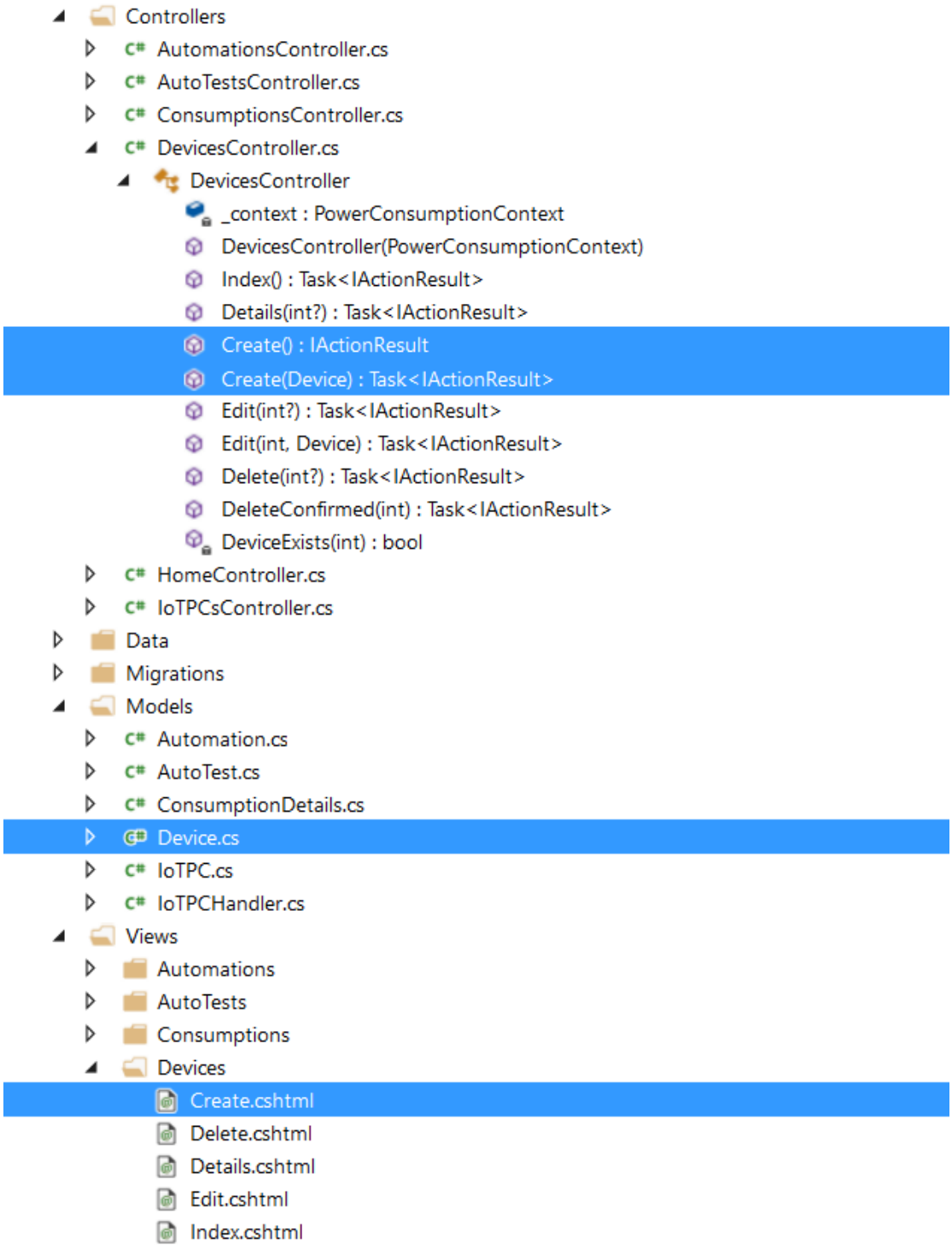
## 6.4 Εφαρμογή Συστήματος

Η διαδικτυακή μας εφαρμογή βασίστηκε στο ASP.NET Core[8] Web Application με δομή MVC, χρησιμοποιώντας δηλαδή την αρχιτεκτονική Model-View-Controller. Τα Models περιέχουν τις οντότητες που συνδέονται στη βάση δεδομένων και τους κανόνες της εφαρμογής, οι controllers το χειρισμό των αλληλεπιδράσεων, δηλαδή τη λογική της λειτουργικότητας και τέλος τα views περιλαμβάνουν τις διεπαφές με το χρήστη, το περιεχόμενο που εμφανίζεται σε αυτόν [52].

### 6.4.1 MVC Template - Αντιστοιχία με την Εφαρμογή

Στη συνέχεια θα δοθεί ένα παράδειγμα χρήσης του μοντέλου MVC, προκειμένου να εξεταστεί η αντιστοιχία της δομής του project με το πρότυπο MVC [53]. Όπως φαίνεται στην εικόνα 6.2, για κάθε στοιχείο του μοντέλου υπάρχει και ο αντίστοιχος φάκελος. Με μπλε χρώμα έχουν επισημανθεί τα περιεχόμενα των φακέλων, τα αρχεία και οι μέθοδοι που θα χρησιμοποιηθούν στο παράδειγμα. Τα ονόματα των φακέλων και των αρχείων έχουν ιδιαίτερη σημασία στην υλοποίηση του MVC από την Microsoft, για τον λόγο αυτό θα μαρκάρονται σύμφωνα με την αλληλεπίδραση που έχουν.

[8] Το NET Core είναι μια πλατφόρμα ανοιχτού κώδικα που υποστηρίζει τα λειτουργικά συστήματα Wndows, Linux, MacOS. Περιέχει πολλά API από το .NET Framework και είναι διαθέσιμο στο github.




**Εικόνα 6-2: MVC template structure**

Συγκεκριμένα:

1. Ο φάκελος Controller περιέχει την κλάση DevicesController.
    a. Η κλάση DevicesController περιέχει δύο υπερφορτωμένες μεθόδους για την ενέργεια Create.

2. Ο φάκελος Model περιέχει την κλάση Device.
3. Ο φάκελος View περιέχει τον φάκελο Devices.
    a. Στον οποίο βρίσκεται το αρχείο Create.cshtml.

Πριν το παράδειγμα, θα πρέπει να εξεταστεί ένα κομμάτι της μεθόδου Configure() που βρίσκεται στο αρχείο Startup.cs.

```
app.UseMvc(
    routes =>
    {
        routes.MapRoute(
            name: "default_route",
            template: "{controller}/{action}/{id?}",
            defaults: new { controller = "Home", action = "Index" });
    }
);
```

Στο κομμάτι κώδικα αυτό δηλώνεται η μορφή που θα έχουν τα url της εφαρμογής και το πώς θα γίνεται η κλήση των αντίστοιχων μερών της. Έτσι, τα url της εφαρμογής έχουν το εξής format:

http://PowerConsumption/ {**controller**}/ {**action**}/ {**id**}

όπου:

{**controller**}: το όνομα του Controller

{**action**}: το όνομα της μεθόδου του Controller

{**id**}: κάποιο προαιρετικό id.

Το όνομα του Controller ορίζεται να είναι το όνομα του αρχείου χωρίς την λέξη «Controller». Για παράδειγμα, το όνομα του αρχείου είναι «DevicesController», αλλά στο url θα υπάρχει μόνο η πρώτη λέξη, δηλαδή «Devices». Επιπλέον ορίζεται ποιος είναι ο προεπιλεγμένος Controller και ποια η προεπιλεγμένη ενέργεια αυτού κατά την εκκίνηση της εφαρμογής, δηλαδή όταν στο url υπάρχει μόνο το https://PowerConsumption/ θα εκτελείται η μέθοδος Index του HomeController.

Σε αυτό το σημείο θα αναφερθούν οι δύο υπερφορτωμένες μέθοδοι με το όνομα Create της κλάσης DevicesController. Οι μέθοδοι των Controllers αντιστοιχούν στα http verbs GET και POST, με κάθε μέθοδο να επιστρέφει ένα View στο τέλος των ενεργειών της. Η αντιστοίχιση της κάθε μεθόδου με το αντίστοιχο verb ορίζεται από τον προγραμματιστή με τα properties, γνωστά και ως annotations που δίνουν επιπλέον ιδιότητες στις μεθόδους.

Έστω ότι ο χρήστης επέλεξε να κάνει εισαγωγή μιας οικιακής συσκευής επιλέγοντας το κατάλληλο link. Τότε στην γραμμή διευθύνσεων θα εμφανιστεί το url:

https://localhost:61366/Devices/Create (ενδεικτικό τοπικό παράδειγμα)

Το αίτημα αυτό του χρήστη, αντιστοιχεί στην εκτέλεση του κώδικα του DevicesController και συγκεκριμένα της μεθόδου Create. Σε αυτή την περίπτωση, επειδή δεν αποστέλλεται κάτι στο server αλλά γίνεται αίτηση περιεχομένου από αυτόν, θα εκτελεστεί η Create που αντιστοιχεί στο verb GET. Με την εκτέλεση της μεθόδου γίνεται η εργασία που έχει οριστεί και στο τέλος η μέθοδος επιστρέφει την κλάση του .Net View. Η κλάση View θα ψάξει στον φάκελο View και θα αναζητήσει τον υποφάκελο Devices. Εκεί θα πρέπει να βρει ένα cshtml αρχείο με όνομα ίδιο με αυτό της μεθόδου προκειμένου να εμφανίσει το περιεχόμενο του στον χρήστη. Στη συνέχεια ο χρήστης καλείται να συμπληρώσει τα πεδία του view και να εκτελέσει το submit action της φόρμας, δηλαδή να στείλει τα δεδομένα στο server. Αυτή η ενέργεια συνεπάγεται στην εκτέλεση της μεθόδου Create του DevicesController που έχει οριστεί για το verb POST. Θα εκτελεστούν οι ενέργειες της μεθόδου, θα αποθηκευτούν τα δεδομένα στη βάση έπειτα από τον έλεγχο εγκυρότητας των στοιχείων εισαγωγής της εγγραφής και θα επιστραφεί η κλάση View, η αναζήτηση της οποίας θα γίνει και πάλι στο φάκελο View και στον υποφάκελο Devices με το όνομα Create.cshtml.

### 6.4.2 Μετρήσεις Κατανάλωσης Ενέργειας

Η εφαρμογή για κάθε χρήστη περιλαμβάνει στοιχεία των τελευταίων μετρήσεων των συσκευών, στοιχεία για τη μηνιαία κατανάλωση ρεύματος, σενάρια χρονοπρογραμματισμού των συνδεδεμένων συσκευών. Για τη διαχείριση όλων αυτών των πληροφοριών έχουν δημιουργηθεί μέθοδοι στους controllers της κάθε οντότητας με τις οποίες έχουν καθοριστεί οι βασικές λειτουργίες CRUD (Create/Read/Update/Delete), έλεγχοι ορθότητας, συγκέντρωση στοιχείων για την

προβολή τους. Παρακάτω αναλύονται κάποιες μέθοδοι που παρουσιάζουν και το μεγαλύτερο ενδιαφέρον σχετικά με το κομμάτι της υλοποίησης των μετρήσεων κατανάλωσης ενέργειας των οικιακών συσκευών.

Η πρώτη μέριμνα για την εφαρμογή μας ήταν η εύρεση ενός τρόπου για τη λήψη των μετρήσεων από τη συσκευή μας και την αποθήκευση τους στη βάση δεδομένων. Έχοντας ήδη ορίσει και επομένως γνωρίζοντας τις παραμέτρους που στέλνονται: (α) μέτρηση της κατανάλωσης ενέργειας της συνδεδεμένης συσκευής (**psum**) (β) σειριακός αριθμός του ελεγκτή (**sn**), δημιουργήθηκε μια GET action για το σκοπό αυτό:

```
// Method that receives the values from the device and posts them to the database


public ActionResult PostData(string sn, double? psum)
{
    var results = "Success";
    var reported = DateTime.Now;


    try
    {
          var iotPC = _context.IoTPCs.FirstOrDefault(c => c.SerialNumber == sn);
          var device = _context.Devices.FirstOrDefault(d => d.IoTPCID == iotPC.ID);


          if (device == null)
          {
              results = "Unknown device";
          }
          else
          {
              if (psum.HasValue && psum.Value != 0)
              {
                  // add consumption
```

```
                    _context.Add(new ConsumptionDetails
                    {
                        DeviceID = device.DeviceID,
                        IoTPCID = iotPC.ID,
                        Psum = psum.Value,
                        CreateTimestamp = reported
                    });
                }
                // save it all
                _context.SaveChanges();
            }
        }
        catch (Exception ex)
        {
           results = "Exception: " + ex.Message;
        }
        return Content(results);
}
```

Στον κώδικα που παρουσιάζεται παραπάνω εντοπίζεται η συσκευή-ελεγκτής (iotPC) με το serialnumber που έχει δοθεί ως παράμετρος στη μέθοδο PostData (επιστρέφεται η εγγραφή στο ερώτημα LINQ), προκειμένου να βρεθεί και η συσκευή οικιακής χρήσης (device) που είναι συνδεδεμένη με το iotPC αυτό. Αν υπάρχει συσκευή και η παράμετρος psum περιέχει μια έγκυρη τιμή μέτρησης, δημιουργείται η εγγραφή στη βάση για τη μέτρηση αυτή και επιστρέφεται η ένδειξη επιτυχίας/αποτυχίας ανάλογα με την κατάσταση του αιτήματος.

**Κύρια σελίδα εμφάνισης των μετρήσεων**

Στην κύρια σελίδα απεικονίζεται ένα γράφημα των μετρήσεων της ημέρας έτσι ώστε ο καταναλωτής να έχει μια σχηματική εικόνα της ημερήσιας κατανάλωσης εντός της

οικίας του. Για την επιλογή του γραφήματος επιλέχθηκαν να χρησιμοποιηθούν google charts.[9]

Υλοποιήθηκε ένα τμήμα scripts στο κάτω μέρος της σελίδας της Index page/view συμπεριλαμβάνοντας τις απαραίτητες κλήσεις μαζί με μια αναφορά window resize προκειμένου το γράφημα να επανασχεδιάζεται κάθε φορά που αλλάζει το μέγεθος του παραθύρου και να είναι responsive.

```
@section Scripts {

    <script                                          type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>

    <script type="text/javascript">

        google.charts.load('current', { 'packages': ['corechart'] });

        google.charts.setOnLoadCallback(function            ()            {
drawChart(@Model.DeviceID); });

        $(window).resize(function () { drawChart(@Model.DeviceID); });

    </script>
```

Η μέθοδος drawChart υλοποιήθηκε στο αρχείο file.js. Η μέθοδος αυτή κάνει μια κλήση AJAX, αφού φορτώσει η σελίδα, για να φέρει τα δεδομένα από μια μέθοδο του controller, να πάρει το αποτέλεσμα JSON και να τα τροφοδοτήσει στο goooogle chart:

```
function drawChart(deviceId) {

    jQuery.ajaxSettings.traditional = true;

    $.ajax(

        {

            url: '/Consumptions/DeviceDay',

            contentType: "application/json; charset=utf-8",

            dataType: "text",

            data: { id: deviceId },

            type: "GET",

            async: false,

            success: function (jsonData) {

                var data = new google.visualization.DataTable(jsonData);
```

[9] Επίσημη ιστοσελίδα: **https://developers.google.com/chart**

```
                var options = { chart: { title: 'Most recent 24 hours of measurements' }, vAxis: { format: '#,##0.0' }};

                var chart = new google.visualization.AreaChart(document.getElementById('chart_div'));

                chart.draw(data, options);

            }

        });

    return false;

}
```

Τα δεδομένα JSON που θα περάσουν στο γράφημα της Google θα πρέπει να είναι σε μορφή αποδεκτή. Τα Google charts χρησιμοποιούν τα δικά τους αντικείμενα DataTable, τα οποία είναι κατά βάση, δισδιάστατοι πίνακες με κάποιους απλούς κανόνες:

- Όλα τα δεδομένα σε κάθε στήλη πρέπει να έχουν τον ίδιο τύπο δεδομένων
- Κάθε στήλη έχει ένα περιγραφικό στοιχείο που περιλαμβάνει τον τύπο δεδομένων της και μια ετικέτα για τη συγκεκριμένη στήλη (προαιρετικά και ένα μοναδικό αναγνωριστικό)
- Τα κελιά του πίνακα μπορούν να λάβουν μεμονωμένες τιμές ή τιμές από ένα μορφοποιημένο string

Ένα δείγμα για το γράφημα αυτού του DataTable ως JSON:

```
{
  "cols": [
     {"label": "Time of Day", "type": "datetime"},
     {"label": "Consumption Watt", "type": "number"}
  ],
  "rows": [
     {"c": [ {"v": "Date(2017,11,15,11,30,0,0)" }, { "v": "880.0" } ] },
     {"c": [ {"v": "Date(2017,11,15,11,31,0,0)" }, { "v": "895.0" } ] },
     {"c": [ {"v": "Date(2017,11,15,11,32,0,0)" }, { "v": "780.5" } ] }
  ]
}
```

Προκειμένου να επιστραφούν τα αποτελέσματα με την παραπάνω μορφή JSON, χρειάστηκε μια δομή δεδομένων που θα την υλοποιήσει. Η κλάση GoogleVizDataTable σχεδιάστηκε για τη διευκόλυνση της δημιουργίας της μορφής αυτής και κατά συνέπεια τη σωστή αναπαράσταση των αποτελεσμάτων.

```
// Class used to facilitate JSON serialization into format required by Google
    public class GoogleVizDataTable
    {
        public IList<Col> cols { get; set; } = new List<Col>();
        public IList<Row> rows { get; set; } = new List<Row>();
        public class Col
        {
            public string label { get; set; }
            public string type { get; set; }
        }
        public class Row
        {
            public IEnumerable<RowValue> c { get; set; }
            public class RowValue
            {
                public object v;
            }
        }
    }
```

Η μέθοδος DeviceDay δέχεται ως παράμετρο το ID της οικιακής συσκευής (device). Αν το ID είναι έγκυρο και έχουν καταγραφεί μετρήσεις για τη συγκεκριμένη συσκευή, τότε ανακτά όλες τις μετρήσεις στην παρεμβαλλόμενη περίοδο και φορτώνει αυτές τις πληροφορίες στο αντικείμενο GoogleVisDataTable και στη συνέχεια τις επιστρέφει σε μορφή JSON.

```
// Return data for a day/24 hours for given device
public IActionResult DeviceDay(int? id)
{
```

```
    // establish an empty table

    var gdataTable = new GoogleVizDataTable();



    gdataTable.cols.Add(new GoogleVizDataTable.Col { label = "Time of Day",
type = "datetime" });

    gdataTable.cols.Add(new GoogleVizDataTable.Col { label = "Consumption
Watt", type = "number" });



    // if ID given is present

    if (id.HasValue)

        // next get the most recent measurement for this device

        var mostRecent = _context.ConsumptionsDetails.Where(d => d.DeviceID
== id.Value)

                    .Select(m       =>       m).OrderByDescending(m       =>
m.CreateTimestamp).Take(1).FirstOrDefault();



        // if we have a recent measurement for this device

        if (mostRecent != null)

        {

            // establish a range of previous to current day/time

            var finish = mostRecent.CreateTimestamp;

            var start = finish.AddDays(-1);



            // fetch a set of measurements for that range

            var recentSet = ConsumptionSetRange(id.Value, start, finish);



            // build out the google datatable using this data

            gdataTable.rows =

                (from set in recentSet

                select new GoogleVizDataTable.Row

                {

                    c = new List<GoogleVizDataTable.Row.RowValue>

                    {
```

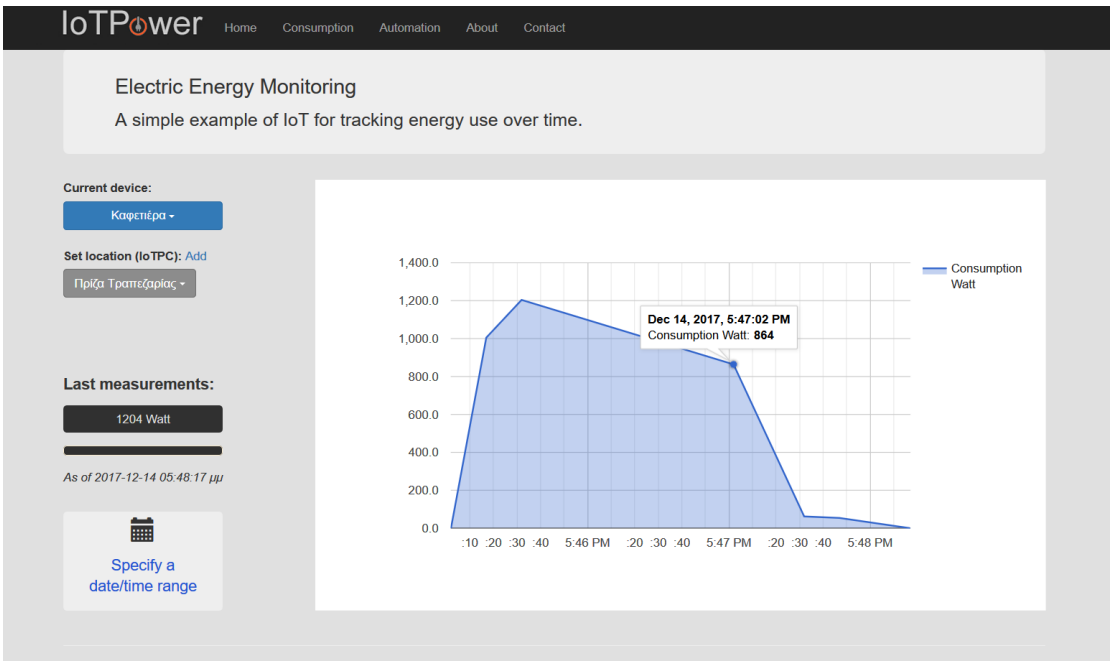
```
                        new GoogleVizDataTable.Row.RowValue { v =
set.GoogleDate },
                        new GoogleVizDataTable.Row.RowValue { v =
set.PowerString }
                    }
                }).ToList();
            }
    }
   return Json(gdataTable);
}
```

Η καθαυτή υλοποίηση της φόρτωσης των μετρήσεων από τη βάση δεδομένων λαμβάνει μέρος σε μια διαφορετική μέθοδο (ConsumptionSetRange) που δέχεται ως παραμέτρους το ID μαζί με τις ημερομηνίες start και end που καθορίζουν και το εύρος της αναζήτησης. Η ιδέα αυτής της μεθόδου έγκειται στη δυνατότητα να μπορούμε να εξάγουμε εκτός από τις ημερήσιες μετρήσεις κατανάλωσης της συσκευής και τις εβδομαδιαίες, μηνιαίες, ετήσιες και γενικώς τις τιμές από οποιουδήποτε εύρος αποφασίσει να ορίσει ο χρήστης.

```
// Build an aggregate list last day's worth of measurements, i.e.
// from the most recent measurement back to 24 hours previous
public List<ConsumptionSet> ConsumptionSetRange(int deviceid, DateTime
start, DateTime finish)
{
    // build the list of measure sets
    var consumptionSet =
        (from c in _context.ConsumptionsDetails
        where c.DeviceID == deviceid
        && c.CreateTimestamp >= start
        && c.CreateTimestamp <= finish
        orderby c.CreateTimestamp
        group c by new { MeasuredDate =
DateTime.Parse(c.CreateTimestamp.ToString("yyyy-MM-dd HH:mm:ss")),
c.Device.IoTPC.Description }
        into g
        select new ConsumptionSet
```

```
        {
            MeasuredDate = g.Key.MeasuredDate,
            LocationName = g.Key.Description,
            ElectricPower = g.Select(c => c.Psum).FirstOrDefault()
        }).ToList();
    return consumptionSet;
}
```

Στην εικόνα 6.3 παρουσιάζεται ένα διάγραμμα ημερήσιας κατανάλωσης της συσκευής που έχει συνδεθεί στον ελεγκτή. Πιο αναλυτικά έχει επιλεχθεί σα συσκευή μια μηχανή του καφέ και από το διάγραμμα απεικονίζεται το φάσμα των τιμών της ηλεκτρικής ενέργειας που καταναλώθηκε στο χρονικό διάστημα της λειτουργίας της συσκευής (5:45–5:49μμ). Παρατηρούμε ότι στο πρώτο λεπτό της έναρξης λειτουργίας σημειώνεται η υψηλότερη τιμή της μέτρησης (περίπου στα 1200 Watt), και μετά από δύο λεπτά η συσκευή μεταβαίνει σε μία κατάσταση ηρεμίας (standby). Θα πρέπει να σημειωθεί επίσης ότι οι μεταβολές στο διάγραμμα εξαρτώνται από τη συχνότητα των μετρήσεων. Στο παράδειγμά μας, στα πλαίσια των εργαστηριακών πειραμάτων οι μετρήσεις γίνονται ανά 15 δευτερόλεπτα.




**Εικόνα 6-3: Ημερήσια Κατανάλωση Επιλεγμένης Συσκευής**

# 7 Επίλογος

## 7.1 Απαιτήσεις Συστήματος – Σύνοψη Λειτουργιών

Ένας πολύ σημαντικός παράγοντας που καθόρισε και την πορεία του έργου ήταν ο σωστός καθορισμός των απαιτήσεων. Συγκεκριμένα, για την παρούσα υλοποίηση, στόχος ήταν η δημιουργία ενός συστήματος για την παρακολούθηση της κατανάλωσης ηλεκτρικής ενέργειας των οικιακών συσκευών καθώς και για τον έλεγχο αυτών, ο οποίος θα μπορούσε να γίνεται είτε απομακρυσμένα είτε τοπικά αλλά κυρίως αυτοματοποιημένα, με απώτερο σκοπό την ελαχιστοποίηση άσκοπων δαπανών ενέργειας. Παρακάτω παρατίθενται αναλυτικά οι απαιτήσεις που εξήχθησαν και ισχύουν τελικώς στο σύστημά μας.

- Το σύστημα παρέχει μηχανισμό διαχείρισης χρηστών. Για τη χρήση του συστήματος απαραίτητη προϋπόθεση αποτελεί η δημιουργία λογαριασμού χρήστη σε αυτό.
- Το σύστημα διαχειρίζεται την καταχώριση των ελεγκτών (IoTPC controllers) μέσω του σειριακού αριθμού που διαθέτουν και την καταχώριση των συσκευών που συνδέονται στους ελεγκτές. Επομένως υπάρχει ο κατάλληλος μηχανισμός για την επιτυχή σύζευξη του συστήματος με τις συσκευές. Επίσης, μέσα από αυτή τη σύνδεση επιτυγχάνεται ο έλεγχος της συσκευής, η αποστολή εντολών από την εφαρμογή προς τις συσκευές. Τέλος, οι συσκευές στέλνουν δεδομένα πίσω στην εφαρμογή σχετικά με την τρέχουσα κατάσταση τους, δηλαδή η εφαρμογή μπορεί να λάβει δεδομένα ανάδρασης.
- Το σύστημα δίνει την δυνατότητα στους χρήστες για την δημιουργία σεναρίων-προγραμμάτων χρονοπρογραμματισμού. Τα προγράμματα ρυθμίζονται από τον χρήστη για την εκτέλεση κάποιας ενέργειας σε πραγματικό χρόνο ή στο μέλλον και με δυνατότητα επιλογής επανάληψης εκτέλεσης. Το πρόγραμμα επενεργεί είτε σε μεμονωμένη συσκευή είτε σε ένα πλήθος συσκευών ταυτόχρονα.
- Ξεκάθαρο και απλό περιβάλλον χρήσης. Το περιβάλλον χρήσης του συστήματος, είναι απλό στη χρήση του. Αυτό σημαίνει πως απαιτούνται εξειδικευμένες γνώσεις και δεξιότητες. Η απλότητα του δε διευκολύνει απλώς τον χρήστη αλλά και τον καθοδηγεί στην εύρεση των λειτουργιών που αναζητεί.

- Η εφαρμογή της ασφάλειας κρίθηκε αναγκαία σε κάθε πτυχή του συστήματος. Αυτό σημαίνει πως η επικοινωνία μεταξύ της εφαρμογής και των συσκευών υποστηρίζει κάποια μορφής ασφάλεια (μηχανισμοί ασφαλείας του MQTT). Ένα ακόμα σημείο στο οποίο δόθηκε έμφαση ήταν η επικοινωνία μεταξύ του χρήστη και της εφαρμογής (μηχανισμοί ασφάλειας του ASP.net framework).
- Επεκτάσιμο και συντηρήσιμο. Οι έννοιες αυτές ορίζουν πως το σύστημα κατασκευάστηκε, και σε επίπεδο κώδικα και σε επίπεδο υλικής κατασκευής, με τρόπο εύκολα κατανοητό από κάποιον τρίτο. Ο κάθε εμπλεκόμενος με το σύστημα μπορεί εύκολα να κατανοεί τη ροή των γεγονότων και να επεμβαίνει όπου χρειάζεται.

## 7.2 Συμπεράσματα - Μελλοντικές Επεκτάσεις

Το σύστημα παρουσιάζει μια εναλλακτική λύση σε ήδη υπάρχουσες υλοποιήσεις αυτοματισμού του IoT στον τομέα των Έξυπνων Οικιών, κυρίως λόγω του χαμηλού κόστους και των χαρακτηριστικών του.

Η κεντρική ιδέα ήταν η δημιουργία ενός συστήματος που θα μπορεί να υποστηρίζει πολλούς χρήστες οικιών χωρίς να περιορίζει τον αριθμό συσκευών που μπορούν να ελεγχθούν. Κάθε συσκευή μπορεί να δρα είτε ανεξάρτητα από τις υπόλοιπες, είτε σαν ομάδα, και για κάθε συσκευή ή ομάδα ο χρήστης μπορεί να εισάγει ένα πλήθος ρυθμίσεων και δυνατοτήτων. Επίσης παρέχεται συγκεντρωτική εποπτεία όλων των συσκευών του χρήστη με πολύ απλό και συμπαγή τρόπο.

Άλλο πλεονέκτημα του συστήματος είναι η επεκτασιμότητα και οι δυνατότητες εύκολης ενσωμάτωσης πρόσθετων λειτουργιών βάσει νέων απαιτήσεων των χρηστών. Σε αυτό συντελεί η επιλογή των αρχιτεκτονικών προτύπων υλοποίησης της πλατφόρμας καθώς και ο ανοιχτού τύπου κώδικας, τόσο σε επίπεδο εφαρμογής, όσο και σε επίπεδο υλικού.

**Επεκτάσεις**

Παρόλο που κατά την υλοποίηση του συστήματος επιτεύχθηκαν οι στόχοι που είχαν τεθεί αρχικά, το συγκεκριμένο σύστημα έχει μεγάλο φάσμα δυνατοτήτων και βελτιώσεων σχετικά με τις παρεχόμενες υπηρεσίες, και αυτό λόγω του χώρου στον οποίο εντάσσεται [54].

Επιμέρους Λειτουργίες του Συστήματος

Μια επέκταση θα μπορούσε να αποτελέσει η ενημέρωση του χρήστη για το ενδεχόμενο υψηλών καταναλώσεων, καθώς και για τους τρόπους άμεσης διερεύνησης των αποκλίσεων για την εξοικονόμηση ενέργειας. Μια λειτουργία εξοικονόμησης σε επίπεδο υλοποίησης θα μπορούσε να παρέχει τη δημιουργία κάποιων σεναρίων αυτόματης ρύθμισης των συσκευών μεμονωμένα ή βάσει επιλεγμένων παραμέτρων ανά κατηγορία.

Σημαντικές υλοποιήσεις θα μπορούσαν να διερευνηθούν και για πιθανούς αστάθμητους παράγοντες που μπορεί να συναντήσουμε σε πραγματικές συνθήκες. Σε περίπτωση μιας ξαφνικής αποσύνδεσης της μονάδας μικροελεγκτή (διακοπή ρεύματος) θα πρέπει με την επαναφορά της σε κατάσταση λειτουργίας, να ειδοποιήσει (ως client) το σύστημά μας για το χρονικό διάστημα στο οποίο υπήρξε απενεργοποίηση. Η ενέργεια αυτή είναι σημαντική προκειμένου να μπορούν να ληφθούν κάποιες αποφάσεις για τις χαμένες εντολές χρονοπρογραμματισμού, για παράδειγμα αν θα συνεχιστεί η εκτέλεσή τους (εφόσον δεν έχει ξεπεραστεί το χρονικό διάστημα για το οποίο είχε οριστεί η εκτέλεση), ή τελικά θα ακυρωθεί.

Η βελτίωση και επέκταση των δυνατοτήτων παρουσίασης των μετρήσεων αποτελεί σημαντική μέριμνα. Θα πρέπει να εξεταστούν υλοποιήσεις, όπως η απεικόνιση του κόστους της ενεργειακής κατανάλωσης με βάση την εκάστοτε χρέωση της κιλοβατώρας από την αντίστοιχη εταιρεία ενέργειας που χρησιμοποιεί ο καταναλωτής. Με αυτόν τον τρόπο ο καταναλωτής θα έχει μια εικόνα της ανάλυσης του ενεργειακού του αποτυπώματος που θα συμβάλλει και στην εξοικονόμηση χρημάτων. Για τη συμβολή αυτή θα μπορούσε να προστεθεί ακόμη η πρόβλεψη της κατανάλωσης μιας επόμενης περιόδου καθώς και μελλοντικές αποκλίσεις λαμβάνοντας υπόψη μετεωρολογικά δεδομένα από API πρόβλεψης καιρού.

Επίσης ιδιαίτερα ενδιαφέρουσα είναι η ιδέα της δυνατότητας σύγκρισης τιμών και προγραμμάτων από διάφορους παρόχους ηλεκτρικής ενέργειας με στόχο την κατάλληλη επιλογή της εταιρείας που θα καλύπτει τις απαιτήσεις του χρήστη με βάση το καταναλωτικό ενεργειακό προφίλ του.

Αρχιτεκτονική υποδομή και θέματα ασφάλειας

Μια επόμενη ερευνητική εργασία σε επίπεδο υλοποίησης θα μπορούσε να είναι η επέκταση της αρχιτεκτονικής του συστήματος με τη χρήση μιας εσωτερικής πύλης (gateway) για τη διαχείριση των επιμέρους ελεγκτών (controllers). Η πύλη αυτή θα είναι υπεύθυνη για τη συλλογή και τον έλεγχο των δεδομένων της οικίας, αποτελώντας τον κεντρικό δίαυλο επικοινωνίας με την πλατφόρμα [55].

Η χρήση του παρόντος συστήματος για να περάσει από το πειραματικό στάδιο σε καταναλωτική εφαρμογή θα απαιτούσε και κάποιες αλλαγές στην υλοποίηση, τόσο στην επιλογή εξαρτημάτων αναφορικά με την ποιότητα και το συνολικό όγκο της διάταξης, όσο και στην εμβάθυνση σε παραμέτρους, όπως ο προσδόκιμος χρόνος λειτουργίας του συστήματος, η ηλεκτρομαγνητική συμβατότητα και θωράκιση.

Τέλος ιδιαίτερα σημαντικό θα ήταν να διερευνηθούν ζητήματα ασφαλείας του δικτύου ως προς θέματα δομής, μεθόδων δρομολόγησης κλπ., και να εξεταστούν οι απειλές οι οποίες είναι πιθανό να θέσουν υπό αμφισβήτηση την ασφάλειά του και οι μηχανισμοί αντιμετώπισής τους [56] [57].

# Βιβλιογραφία


[1] R. Minerva, A. Biru, D. Rotondi, "Towards a definition of the Internet of Things (IoT)," IEEE Internet Initiative, Torino, Italy, 2015.

[2] "Radio-Frequency Identification." Wikipedia, the Free Encyclopedia, https://en.wikipedia.org/wiki/Radiofrequency_identification

[3] C. Pereira and A. Aguiar, "Towards Efficient Mobile M2M Communications: Survey and Open Challenges," Sensors, vol. 14, pp. 19582-19608, 2014.

[4] C. Saad, B. Mostafa, E. Ahmadi and H. Abderrahmane, "Comparative Performance Analysis of Wireless Communication Protocols for Intelligent Sensors and Their Applications", International Journal of Advanced Computer Science and Applications (IJACSA), vol. 5, no. 4, pp. 76-85, 2014.

[5] T. Mendes, R. Godina, E. Rodrigues, J. Matias and J. Catalão, "Smart and energy-efficient home implementation: Wireless communication technologies role," in 5th International Conference on Power Engineering, Energy and Electrical Drives (POWERENG), pp. 377-382, 2015.

[6] O. Horyachyy, "Comparison of Wireless Communication Technologies used in a Smart Home: Analysis of wireless sensor node based on Arduino in home automation scenario," Faculty of Computing at Blekinge Institute of Technology, July 03, 2017.

[7] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6), Specification." Internet Engineering Task Force (IETF) RFC-2460, 1998. Retrieved from https://tools.ietf.org/html/rfc2460

[8] W. Colitti, K. Steenhaut, N. D. Caro, B. Buta and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks," Local & Metropolitan Area Networks, 2011.

[9] CoRE group, IETF, "Charter for CoRE Working Group," IETF. Available: datatracker.ietf.org/wg/core/charter/

[10] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF) RFC-7252, 2014. Retrieved from https://tools.ietf.org/search/rfc7252

[11] "MQTT Version 3.1.1 - OASIS Standard", OASIS, October 2014. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[12] "MQTT and CoAP, IoT Protocols", Eclipse, February 2014. Available: https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2

[13] C.Alippi, Intelligence for Embedded Systems. Springer, 2014.

[14] Κ. Πεκμεστζή, Συστήματα Μικροϋπολογιστών ΙΙ, Συμμετρία, 2009

[15] Dhananjay V. Garde, Programming and Customizing the AVR Microcontroller, 2001 ARM Architecture Reference Manual, ARM Limited

[16] R. Malaric, Instrumentation and Measurement in Electrical Engineering, BrownWalker Press.2011.

[17] S. Ganesan, Selection of current transformers and wire sizing in substations, 2012

[18] Isolated Current and Voltage Transducers Characteristics - Applications – Calculations, LEM Coporate Communications, 1996

[19] Gerard Meijer, Smart Sensors Systems, Wiley, 2008

[20] "Power in AC Circuits." Available: https://www.electronics-tutorials.ws/accircuits/power-in-ac-circuits.html

[21] Principles of Data Acquisition and Conversion, Texas Instruments Inc., Rev. 2015

[22] R.V.D Plass, Integrated Analog-To-Digital and Digital-To-Analog Converters, Springer, 1994

[23] M. Alaa, A.A. Zaidan, B.B. Zaidan, M. Talal, M.L.M. Kiah, "A review of smart home applications based on Internet of Things," Journal of Network and Computer Applications, 97:48–65, 2017.

[24] M. Cabras M, V. Pilloni V, "A novel Smart Home Energy Management system: Cooperative neighbourhood and adaptive renewable energy usage. Communications." IEEE International Conference on Data Science and Data Intensive Systems, 2015.

[25] M. Simon, V. Ruben, K. Matthias, and M. Friedemann, "Smart Configuration of Smart Environments," IEEE Transactions on Automation Science and Engineering, 2016.

[26] "Home and building automation system" Available: http://www.gavazzi-automation.com/docs/download_area/HOME_AUTO_SYS.pdf

[27] "Ενημερωτικός οδηγός instabus KNX", Available: https://w5.siemens.com/greece/internet/el/pss/IC/BT/eit/Documents/INSTABUS KNX2012.pdf

[28] D-Link website. Available: https://eu.dlink.com/gr/el/for-home/smart-home

[29] Smartwatt website. Available: https://www.smartwatt.gr/el

[30] L. Salman, S. Salman, S. Jahangirian, M. Abraham, F. German, C. Blair, & P. Krenz, "Energy efficient IoT-based smart home," 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, Dec. 12-14, 2016.

[31] S. Balamurugan, D. Saravanakamalam, "Energy Monitoring and Management using Internet of Things," IEEE, International Conference on Power and Embedded Drive Control, 2017.

[32] B.S. Brad, M. M. Murar, "Smart Buildings Using IoT Technologies".

[33] C. Hao, & T. Yoshimura, "Economical smart home scheduling for single and multiple users," presented at 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), Oct. 16-19, 2016.

[34] "C Sharp Programming." WikiBooks, the open-content textbooks collection. Available: http://en.wikibooks.org/wiki/C_Sharp_Programming

[35] ".NET." https://www.microsoft.com/net

[36] "What is Entity Framework?" http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[37] V. P. Mehta, Pro LINQ Object Relational Mapping with C# 2008, Apress, 2008

[38] "Arduino" Wikipedia, the Free Encyclopedia. https://el.wikipedia.org/wiki/Arduino

[39] M. Banzi, "Getting Started with Arduino, Second Edition", O'Reilly Media, 2011

[40] "Arduino Uno R3" https://store.arduino.cc/arduino-uno-rev3

[41] "D1 mini [Wemos Electronics]" https://wiki.wemos.cc/products:d1:d1_mini

[42] "Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor." Available: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

[43] "SCT-013 Datasheet - MCI Electronics", Available: https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_SCT013.pdf

[44] M. Margolis, "Arduino Cookbook, Second Edition", O'Reilly Media, 2011

[45] A.S. Sedra, K.C. Smith, Μικροηλεκτρονικά Κυκλώματα, Παπασωτηρίου, 5η έκδοση, 2010

[46] "ACS712 Arduino AC Current Tutorial" Available online at: http://henrysbench.capnfatz.com/henrys-bench/arduino-current-measurements/acs712-arduino-ac-current-tutorial

[47] "RMS Voltage Tutorial" Available online at: https://www.electronics-tutorials.ws/accircuits/rms-voltage.html

[48] Arduino Language Reference https://www.arduino.cc/reference

[49] "IoT with an ESP8266 (Part2) – Arduino Sketch", Available online at: https://www.intertech.com/Blog/iot-with-an-esp8266-part-2-arduino-sketch

[50] A. Dorri, S.S. Kanhere, R. Jurdak, & P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," presented at IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), March 13-17, 2017.

[51] Getting Started with Entity Framework 6 Code First using MVC 5 Tom Dykstra, Rick Anderson, E-book publication date: April, 2014

[52] ASP.NET MVC 6 Documentation Release Microsoft March 02, 2016

[53] "Get started with ASP.NET Core MVC and Visual Studio" Available: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc

[54] H. Ghayvat, S.C Nukhopadhyay, J. Liu, A. Babu, Md E. Elahi, X. Gui, "Internet of Things for smart homes and buildings, Opportunities and Challenges".

[55] C. Stergiou, K.E. Psannis, A.P. Plageras, G. Kokkonis, & Y. Ishibashi, "Architecture for security monitoring in IoT environments," presented at IEEE 26th International Symposium on Industrial Electronics (ISIE), June 19-21, 2017.

[56] P.A. Plageras, K.E. Psannis, B. Gupta, C. Stergiou, B. Kim, & Y. Ishibashi, "Solutions for inter-connectivity and security in a smart hospital building", IEEE 15th International Conference on Industrial Informatics (INDIN), July 24-26, 2017.

[57] C. Stergiou, K.E. Psannis, B. Kim, & B. Gupta, "Secure integration of IoT and Cloud Computing," in Future Generation Computer Systems, 78, 964-975, 2016.