Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: Εφαρμογή Οικιακού Αυτοματισμού με Απομακρυσμένο Έλεγχο μέσω Κινητού και Διαδικτύου

Φραγκιαδάκης Αδάμ (ΑΜ: 3273)

Επιβλέπων εκπαιδευτικός: Παναγιωτάκης Σπυρίδων

Ημερομηνία παρουσίασης: 16/02/2018

Ευχαριστίες:

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω πρώτα τον απόφοιτο του τμήματος Ηλεκτρολογίας του ΤΕΙ Κρήτης, Μανώλη Κατσαγαράκη για την σημαντική βοήθεια που μου προσέφερε στην υλοποίηση του κυκλώματος ελέγχου της φωτεινότητας του λαμπτήρα (dimmer) και τον επιβλέπων καθηγητή μου κ. Σπύρο Παναγιωτάκη για την άμεση βοήθεια που μου έδινε στα προβλήματα που προκύπταν στην πορεία της εργασίας αλλά και για τις συστάσεις του.

Επίσης, θα ήθελα να ευχαριστήσω τους καθηγητές του τμήματος για την γνώση και την τεχνογνωσία που μου μετέδωσαν όσον αφορά τις σύγχρονες ανάγκες που έχει η κοινωνία από την επιστήμη της πληροφορικής και τις εφαρμογές της, καθώς και τα άτομα που με βοήθησαν με τον ορθογραφικό και συντακτικό έλεγχο της παρούσας αναφοράς.

Abstract:

In this thesis, a system of home automation is implemented, based on the open source platform openHAB 2. For the implementation of this system, a wireless network consisted of devices and sensors was deployed with the central node and server being a microcomputer Raspberry PI 3. The microcontrollers ESP8266 – 12 – f and the board ESP8266 – 12 – EVB are used as clients. On these microcontrollers are connected the device control circuits. The communication in the network is being achieved with the use of the machine – to – machine protocol, MQTT. The purpose of this network is the remote control of the home devices / appliances via a smartphone or a personal computer.

Σύνοψη:

Στην παρούσα πτυχιακή εργασία υλοποιείται ένα σύστημα οικιακού αυτοματισμού βασισμένο πάνω στην ανοιχτού πηγαίου κώδικα πλατφόρμα openHAB 2. Για την υλοποίηση του συστήματος αυτού δημιουργήθηκε ένα ασύρματο δίκτυο συσκευών και αισθητήρων με κεντρικό κόμβο και ρόλο server έναν μικροϋπολογιστή Raspberry PI 3. Στον ρόλο του client υπάρχουν οι μικροελεγκτές ESP8266 – 12 – f και η πλακέτα ESP8266 – EVB, πάνω στους οποίους είναι συνδεδεμένα τα κυκλώματα ελέγχου των συσκευών. Η επικοινωνίας μέσα στο δίκτυο επιτυγχάνεται με την χρήση του machine – to – machine πρωτοκόλλου MQTT. Ο σκοπός του δικτύου αυτού είναι ο απομακρυσμένος έλεγχος των συσκευών του σπιτιού μέσω ενός κινητού τηλεφώνου ή ενός ηλεκτρονικού υπολογιστή.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

1.1 Περίληψη

Σήμερα, το Internet of Things πιάνει ένα όλο και μεγαλύτερο κομμάτι στην αγορά των τεχνολογιών καθώς προσφέρει λύσεις και ευκολία στην καθημερινότητα ενός ανθρώπου. Ένα απλό δίκτυο αισθητήρων μπορεί να προσφέρει λύσεις από την γεωργία και την απομακρυσμένη παρακολούθηση μιας μονάδας παραγωγής μέχρι και την ιατρική. Επιπλέον, με τα smartphones να αποτελούν, πλέον, αναπόσπαστο κομμάτι της ζωής μας η διαχείριση και η παρακολούθηση τέτοιων συστημάτων έχει γίνει πιο εύκολη και προσιτή από ποτέ.

Οι οικιακοί αυτοματισμοί είναι ένα μέρος του Internet of Things που ανήκει στην κατηγορία των εφαρμογών για καταναλωτές και είναι συνεχώς σε εξέλιξη με νέες πλατφόρμες, τεχνολογίες και hardware να βγαίνουν κάθε τόσο. Ένας οικιακός αυτοματισμός μπορεί να προσφέρει οικονομία καθώς κάνει την παρακολούθηση κατανάλωσης ενέργειας πιο εύκολη από ποτέ, ασφάλεια με την εγκατάσταση ενός δικτύου αισθητήρων και κάμερας και ευκολότερη εξ' αποστάσεως διαχείριση όλων των οικιακών συσκευών.

Στην παρούσα πτυχιακή εργασία υλοποιείται ένα smart home δίκτυο για την διαχείριση συμβατικών και smart ηλεκτρολογικών συσκευών σε συνδυασμό με ένα σύστημα ασφαλείας. Η υλοποίηση προσφέρει επίσης ευφυής αυτοματισμούς καθώς στην αλλαγή της κατάστασης κάποιων αντικειμένων ενεργοποιούνται άλλοι μηχανισμοί του συστήματος μέσω της μηχανής κανόνων. Το σύστημα αυτό με τις κατάλληλες τροποποιήσεις και χρήση αισθητήρων μπορεί να χρησιμοποιηθεί και για άλλες εφαρμογές αυτοματισμού στο πλαίσιο του Internet of Things όπως για παράδειγμα, εφαρμογές τηλεϊατρικής ή γεωργίας.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Τα τελευταία χρόνια, λόγω της εκρηκτικής ανάπτυξης των συσκευών που συνδέονται στο διαδίκτυο και ελέγχονται μέσω αυτού, έχει οδηγήσει το Internet of Things να είναι αρκετά δημοφιλές με διάφορες εφαρμογές όπως φαίνεται και παρακάτω στην εικόνα 1 – 1. Όλο και περισσότερες επιχειρήσεις έχουν αρχίσει να υιοθετούν τις εφαρμογές του αλλά και να πουλούν τεχνολογικά προϊόντα σχετικά με το Internet of Things για χρήση από καταναλωτές. Παρ' όλα αυτά αν κάποιος χρήστης θέλει να στήσει ένα ολοκληρωμένο σύστημα οικιακού αυτοματισμού από τα προϊόντα αυτά θα χρειαστεί να αλλάξει σχεδόν όλες τις οικιακές συσκευές του και για κάθε μία από αυτές να έχει διαφορετική εφαρμογή.

Για αυτό τον λόγο σε αυτή την εργασία αναπτύσσεται ένα σύστημα αυτοματισμού που εισάγει τις συμβατικές συσκευές στον smart κόσμο με την χρήση κατάλληλων μικροελεγκτών και αισθητήρων και σε συνδυασμό με τις smart συσκευές είναι διαχειρίσιμο από μία εφαρμογή. Τέλος, το χαμηλό κόστος, η ευελιξία και η ζήτηση ενός τέτοιου συστήματος αποτέλεσαν επιπλέον παράγοντες για την διεξαγωγή της εργασίας

1.3 Σκοπός και Στόχοι Εργασίας

Σκοπός της εργασίας είναι η δημιουργία ενός ευέλικτου συστήματος με χαμηλό κόστος για την απομακρυσμένη διαχείριση των συσκευών μίας οικιακής εγκατάστασης από ένα smartphone ή έναν ηλεκτρονικό υπολογιστή καθώς και την ανάπτυξη ενός βασικού συστήματος ασφαλείας. Εφ' όσον το δίκτυο των συσκευών μας είναι συνδεδεμένο με το διαδίκτυο, τότε ο εκάστοτε χρήστης θα έχει την δυνατότητα να παρακολουθεί την κατάσταση του σπιτιού αλλά και να αλληλοεπιδρά με τις συσκευές οποιαδήποτε στιγμή, οπουδήποτε στον κόσμο.

1.4 Δομή Εργασίας

Στην παρούσα εργασία αναλύονται οι τεχνολογίες του Internet of Things και των ασυρμάτων δικτύων αισθητήρων, οι οποίες αποτελούν την βάση για την υλοποίηση ενός οικιακού αυτοματισμού. Στο κεφάλαιο 3 περιγράφεται η πλατφόρμα openHAB και πως χρησιμοποιείται στην παρούσα υλοποίηση και παρακάτω στο κεφάλαιο 4 περιγράφεται λεπτομερώς βήμα προς βήμα το εργαστηριακό κομμάτι της εργασίας. Εν συνεχεία, στο κεφάλαιο 5 αναφέρονται τα σενάρια λειτουργείας του οικιακού αυτοματισμού σε πραγματικό περιβάλλον. Τέλος στο 6ο κεφάλαιο περιγράφονται οι μελλοντικές επεκτάσεις που θα μπορούσε να έχει η εφαρμογή, συμπεράσματα που αφορούν όλη την πτυχιακή εργασία και μία συνολική αξιολόγησή της. Συνοπτικά η δομή της εργασίας είναι η εξής:

- Κεφάλαιο 2: περιγραφή του Internet of Things και των τεχνολογιών του, περιγραφή και χρήση ασύρματων δικτύων αισθητήρων.
- Κεφάλαιο 3: Περιγραφή openHAB και τρόπος λειτουργείας του για την υλοποίηση ενός οικιακού αυτοματισμού.
- Κεφάλαιο 4: Περιγραφή επιθυμητού δικτύου, ανάλυση hardware και software, περιγραφή υλοποίησης συστήματος.
- Κεφάλαιο 5: παρουσίαση σεναρίων λειτουργείας του συστήματος με εικόνες
- Κεφάλαιο 6: Συμπεράσματα, μελλοντικές επεκτάσεις του συστήματος, αναφορά μειονεκτημάτων και συνολική αξιολόγηση της εργασίας

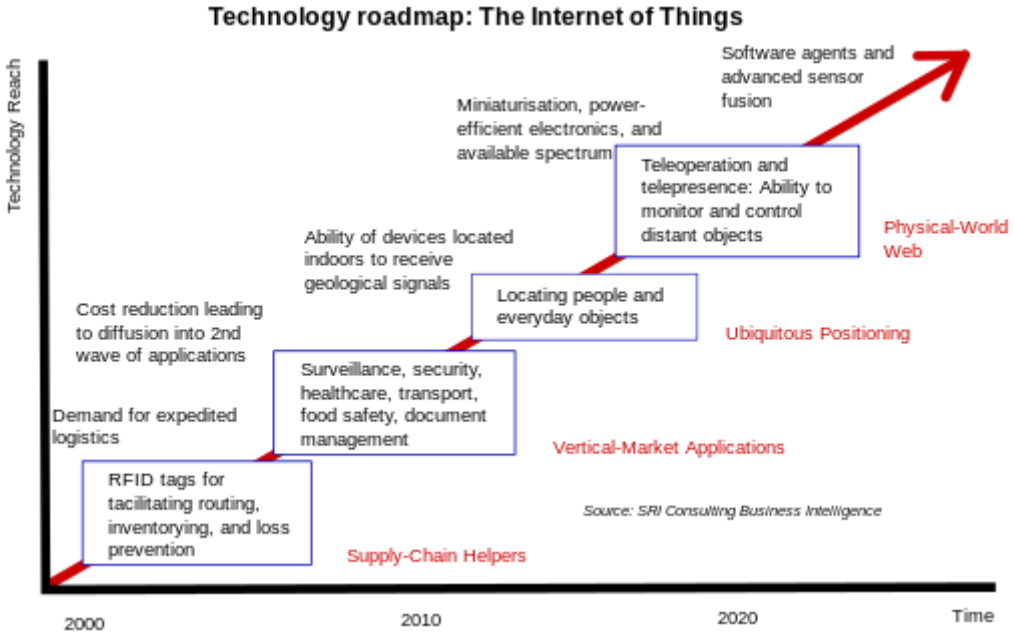

Figure 1-1 Εφαρμογές IoT

2 Internet of Things

Figure 2-1 Internet of Things

2.1 Τι είναι

Το Internet of Things (IoT για συντομία) ή Διαδίκτυο των Πραγμάτων είναι ένα δίκτυο από φυσικές συσκευές, οχήματα, οικιακές συσκευές και άλλων αντικειμένων που είναι ενσωματωμένα με ηλεκτρονικά, λογισμικό, αισθητήρες, ενεργοποιητές και έχουν τη δυνατότητα σύνδεσης σε κάποιο δίκτυο με αποτέλεσμα αυτά να μπορούν να συνδεθούν και να ανταλλάξουν δεδομένα μεταξύ τους. Κάθε αντικείμενο είναι μοναδικά αναγνωρίσιμο μέσω του ενσωματωμένου υπολογιστικού συστήματός του και είναι ικανό να αλληλεπιδρά εντός της υπάρχουσας υποδομής του Internet. Ειδικοί εκτιμούν ότι το Internet of Things θα αποτελείται από περίπου 30 δισεκατομμύρια συσκευές μέχρι το 2020. Μέχρι το ίδιο έτος εκτιμάται επίσης ότι η αξία του στην παγκόσμια αγορά θα φτάσει τα 7.1 τρισεκατομμύρια δολάρια.

Το Internet of Things επιτρέπει σε αντικείμενα να είναι αισθητά και να ελέγχονται απομακρυσμένα μέσω υπάρχουσας υποδομής δικτύων, δημιουργώντας ευκαιρίες για πιο άμεση ενσωμάτωση του φυσικού κόσμου μέσα σε συστήματα βασισμένα σε υπολογιστές. Αυτό έχει ως αποτέλεσμα την βελτιωμένη αποδοτικότητα, ακρίβεια και οικονομικό όφελος, λαμβάνοντας υπ' όψιν την μειωμένη ανθρώπινη παρέμβαση. Όταν το Internet of Things είναι ενισχυμένο με αισθητήρες και ενεργοποιητές, η τεχνολογία γίνεται ένα υπόδειγμα της πιο γενικής κλάσης των cyber-physical systems, όπου περιλαμβάνει τεχνολογίες όπως smart grids, virtual power plants, smart homes, intelligent transportation και smart cities.

Ο όρος «πράγματα» στην έννοια του Internet of Things, μπορεί να αναφέρεται σε μία μεγάλη ποικιλία συσκευών όπως εμφυτεύματα για την παρακολούθηση της καρδιάς, biochip αναμεταδότες σε εκτρεφόμενα ζώα, αυτοκίνητα με ενσωματωμένους αισθητήρες ή συσκευές πεδίου που βοηθούν πυροσβέστες ή ομάδες σε επιχειρήσεις εύρεσης ανθρώπων κ.α. Επιστήμονες προτείνουν να θεωρούμε τα «πράγματα» σαν «αδιάλυτο μίγμα hardware, software, δεδομένων και υπηρεσιών». Αυτές οι συσκευές, λοιπόν, συλλέγουν χρήσιμα δεδομένα με την βοήθεια διάφορων τεχνολογιών και μετά, αυτόνομα, δημιουργούν μία ροή δεδομένων μεταξύ τους.

Ο όρος Internet of Things επινοήθηκε στα τέλη της δεκαετίας στου 1990 από τον επιχειρηματία Kevin Ashton. Ο Ashton, ο οποίος είναι ένας από τους ιδρυτές του Auto-ID Center στο MIT, ήταν μέρος μιας ομάδας που ανακάλυψε τον τρόπο να συνδέσει τα αντικείμενα με το διαδίκτυο μέσω μιας ετικέτα RFID. Έχει δηλώσει ότι χρησιμοποίησε πρώτη φορά τη φράση "Internet of Things" σε μια παρουσίαση που έκανε το 1999 και ο όρος αυτός έχει κολλήσει από τότε.

2.2 Χαρακτηριστικά

Η σημαντική δημοσιότητα που έχει λάβει το Internet of Things τα τελευταία χρόνια οφείλεται στην εκρηκτική ανάπτυξη συσκευών που μπορούν να συνδεθούν και να εκλεχθούν

από το διαδίκτυο. Το μεγάλο φάσμα των εφαρμογών της τεχνολογίας του Internet of Things σημαίνει πως τα χαρακτηριστικά μπορούν να διαφέρουν από συσκευή σε συσκευή. Παρ ‘ όλα αυτά υπάρχουν κάποια βασικά χαρακτηριστικά που εκφράζουν τις περισσότερες από αυτές τις συσκευές. Τα 7 αυτά βασικά χαρακτηριστικά είναι τα εξής:

1. **Νοημοσύνη**:

Το Internet of Things προέρχεται από ένα συνδυασμό από αλγόριθμους και υπολογισμούς, software και hardware, που το κάνουν έξυπνο. Η περιβαλλοντική νοημοσύνη στο Internet of Things ενισχύει τις δυνατότητες του οι οποίες διευκολύνουν τα αντικείμενα να ανταποκρίνονται με ένα νοήμων τρόπο σε μία συγκεκριμένη κατάσταση και τα υποστηρίζει να φέρνουν εις πέρας συγκεκριμένα καθήκοντα. Παρ’ όλη την δημοσιότητα των έξυπνων τεχνολογιών, η νοημοσύνη στο Internet of Things αποτελεί ένα μέσο διάδρασης μεταξύ των συσκευών, ενώ η διάδραση χρήστη – συσκευής είναι επιτευκτή από standard input μεθόδους και διεπαφής.

2. **Ανομοιογένεια**:

Η ανομοιογένεια στο Internet of Things αποτελεί ένα από τα βασικά «κλειδιά» χαρακτηριστικά. Οι Συσκευές στο IoT είναι βασισμένες σε διαφορετικές πλατφόρμες hardware και δικτύων και μπορούν να διαδράσουν με άλλες συσκευές ή πλατφόρμες υπηρεσιών μέσω άλλων δικτύων. Η αρχιτεκτονική του Internet of Things θα έπρεπε να υποστηρίζει άμεση δικτυακή συνδεσιμότητα μεταξύ ετερογενών δικτύων. Οι σχεδιαστικές απαιτήσεις κλειδί για τα ετερογενή αντικείμενα και τα περιβάλλοντά τους στο IoT είναι οι κλιμάκωση, modularity, επεκτασιμότητα και διαλειτουργικότητα.

3. **Δυναμική φύση:**

Πρωτεύουσα δραστηριότητα του Internet of Things είναι να συλλέγει δεδομένα από το περιβάλλον του, αυτό είναι επιτευκτό με τις δυναμικές αλλαγές που λαμβάνουν χώρα στις συσκευές. Η κατάσταση αυτών των συσκευών αλλάζει δυναμικά, για παράδειγμα sleeping και waking up, connected και/ή disconnected, όπως επίσης και το πλαίσιο των συσκευών συμπεριλαμβανομένης και της θερμοκρασίας, τοποθεσίας και ταχύτητας της συσκευής. Επιπροσθέτως, ο αριθμός των συσκευών επίσης μπορεί να αλλάξει δυναμικά.

4. **Ανίχνευση**:

Το Internet of Things θα ήταν αδύνατο χωρίς την χρήση αισθητήρων οι οποίοι θα εντοπίσουν ή θα μετρήσουν οποιαδήποτε αλλαγή στο περιβάλλον για να δημιουργήσουν δεδομένα που μπορούν να αναφέρονται στην κατάστασή του ή ακόμα και να διαδρούν με αυτό. Οι τεχνολογίες αίσθησης παρέχουν τα μέσα για την δημιουργία ικανοτήτων που ανακλούν μία πραγματική επίγνωση του φυσικού κόσμου και των ανθρώπων σε αυτόν. Οι πληροφορίες από τους αισθητήρες είναι απλά η αναλογική είσοδος από τον φυσικό κόσμο, αλλά μπορεί να παρέχει την πλούσια κατανόηση του πολύπλοκου κόσμου μας.

5. **Μεγάλη κλίμακα**:

Ο αριθμός των συσκευών που χρειάζεται να διαχειρισθεί και να επικοινωνούν μεταξύ τους θα είναι τουλάχιστον μία τάξη μεγέθους μεγαλύτερος από τις συσκευές που είναι συνδεδεμένες στο τρέχον διαδίκτυο. Ακόμα πιο καίρια θα είναι η διαχείριση των δεδομένων που δημιουργούνται και η ερμηνεία τους για σκοπούς εφαρμογής. Αυτό σχετίζεται τόσο στην σημασιολογία των δεδομένων, όσο και στην αποδοτική διαχείριση αυτών.

6. **Συνδεσιμότητα**:

Η συνδεσιμότητα ενδυναμώνει το Internet of Things φέρνοντας μαζί καθημερινά αντικείμενα. Η συνδεσιμότητα αυτών των αντικειμένων είναι ζωτικής σημασίας επειδή απλές

διαδράσεις σε επίπεδο αντικειμένου συνεισφέρουν προς την συλλογική νοημοσύνη σε ένα IoT δίκτυο. Ενεργοποιεί την δικτυακή προσβασιμότητα και συμβατότητα στα αντικείμενα. Με αυτήν την συνδεσιμότητα, νέες ευκαιρίες αγοράς για το Internet of Things μπορούν να δημιουργηθούν από την δικτύωση έξυπνων συσκευών και εφαρμογών.

7. **Ασφάλεια**:

Οι συσκευές του Internet of Things είναι από την φύση τους ευάλωτες σε απειλές ασφαλείας. Όσο κερδίζουμε αποτελεσματικότητα, νέες εμπειρίες και άλλα πλεονεκτήματα από το IoT, θα ήταν λάθος να ξεχάσουμε θέματα σχετικά με την ασφάλεια. Υπάρχει ένα υψηλό επίπεδο θεμάτων διαφάνειας και ιδιωτικότητας στο IoT. Είναι σημαντικό να ασφαλίζουμε τα τελικά σημεία, τα δίκτυα και τα δεδομένα που μεταφέρονται. Όλο αυτό σημαίνει την δημιουργία ενός συστήματος ασφαλείας.

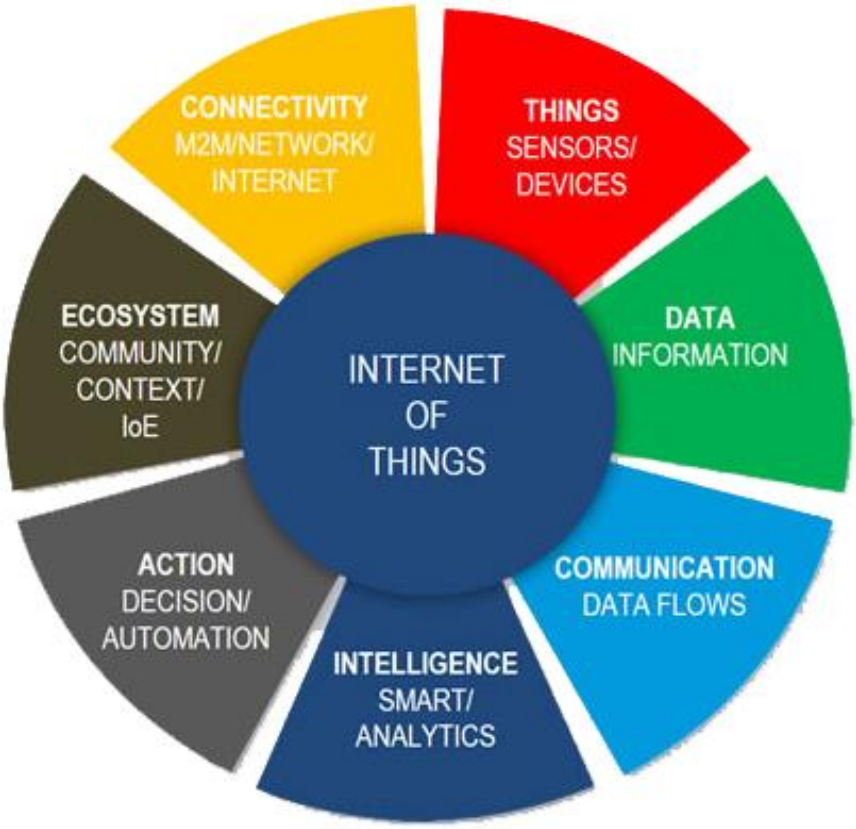

Figure 2-2 IoT characteristics

Υπάρχει μία μεγάλη ποικιλία τεχνολογιών που σχετίζονται με το Internet of Things που διευκολύνουν στην ορθή του λειτουργεία. Οι IoT τεχνολογίες κατέχουν τα προαναφερθέντα χαρακτηριστικά τα οποία δημιουργούν αξία και υποστηρίζουν τις ανθρώπινες δραστηριότητες· επιπλέον ενισχύουν τις δυνατότητες του IoT δικτύου από την αμοιβαία συνεργασία και με το να γίνονται κομμάτι της ολότητας του συστήματος.

2.3 Αρχιτεκτονική

Η αρχιτεκτονική του Internet of Things αποτελείται από διαφορετικά επίπεδα τεχνολογιών που το υποστηρίζουν. Η αρχιτεκτονική εξυπηρετεί στην απεικόνιση του πως διάφορες τεχνολογίες σχετίζονται μεταξύ τους και για να δώσει να καταλάβουμε την κλιμάκωση, την σπονδυλότητα και την διαμόρφωση των IoT αναπτύξεων σε διαφορετικά σενάρια. Η αλήθεια, βέβαια, είναι πως δεν υπάρχει κοινό πρότυπο στην αρχιτεκτονική του IoT που να συμφωνούν όλοι παγκόσμια και διάφορες αρχιτεκτονικές έχουν προταθεί από διάφορους ερευνητές. Η πιο βασική όμως, η οποία αποτελείται από 4 επίπεδα, περιγράφεται παρακάτω και απεικονίζεται στην εικόνα 2 – 3.

2.3.1 Έξυπνη συσκευή / sensor layer:

Το χαμηλότερο επίπεδο αποτελείται από έξυπνα αντικείμενα ενσωματωμένα με αισθητήρες. Οι αισθητήρες ενεργοποιούν την διασυνδεσιμότητα του φυσικού και ψηφιακού κόσμου επιτρέποντας σε πραγματικό χρόνο την συλλογή και επεξεργασία πληροφοριών. Υπάρχουν διάφοροι τύποι αισθητήρων για διάφορους σκοπούς. Οι αισθητήρες έχουν την ικανότητα να κάνουν μετρήσεις για παράδειγμα, θερμοκρασίας, ποιότητας αέρα, ταχύτητας, υγρασίας, ατμοσφαιρικής πίεσης, ροής, κίνησης, ηλεκτρισμού κ.α. Σε ορισμένες περιπτώσεις μπορούν επίσης να διαθέτουν, έως ένα βαθμό, μνήμη, δίνοντάς τους την δυνατότητα να καταγράφουν έναν ορισμένο αριθμό μετρήσεων. Ένας αισθητήρας μπορεί να μετρήσεις την φυσική ιδιότητα και να την μετατρέψει σε σήμα που μπορεί να κατανοηθεί από κάποιο εργαλείο. Οι αισθητήρες είναι ομαδοποιημένοι ανάλογα με τον μοναδικό τους σκοπό, όπως περιβαλλοντικοί αισθητήρες, αισθητήρες σώματος, αισθητήρες οικιακών εφαρμογών, αισθητήρες οχημάτων τηλεματικής κ.α.

Οι περισσότεροι αισθητήρες απαιτούν σύνδεση με τις πύλες (gateways) αισθητήρων. Αυτό μπορεί να έχει την μορφή ενός Local Area Network (LAN) όπως ethernet και Wi – Fi συνδέσεις ή Personal Area Network (PAN) όπως ZigBee, Bluetooth και Ultra – Wideband (UWB). Για τους αισθητήρες που δεν απαιτείται συνδεσιμότητα σε συγκεντρωτές αισθητήρων, η συνδεσιμότητά τους σε backend servers/applications μπορεί να επιτευχθεί χρησιμοποιώντας Wide Area Network (WAN) όπως GSM, GPRS και LTE. Αισθητήρες που χρησιμοποιούν χαμηλής ισχύς και χαμηλό ρυθμό δεδομένων συνδεσιμότητα, τυπικά σχηματίζουν δίκτυα κοινώς γνωστά σαν wireless sensor networks (WSNs). Τα WSNs αποκτούν δημοσιότητα καθώς μπορούν να φιλοξενήσουν πολύ περισσότερους κόμβους αισθητήρων ενώ διατηρούν επαρκή ζωή μπαταρίας και καλύπτουν μεγάλες περιοχές.

2.3.2 Gateways and Networks:

Καθώς τεράστιος όγκος δεδομένων θα τεθεί υπό επεξεργασία από αυτούς τους μικροσκοπικούς αισθητήρες, απαιτείται μία εύρωστη και υψηλής απόδοσης ενσύρματη ή ασύρματη υποδομή δικτύου να λειτουργεί σαν μέσο μεταφοράς. Τα υπάρχοντα δίκτυα, τα οποία είναι συχνά δεσμευμένα με διαφορετικά πρωτόκολλα, έχουν χρησιμοποιηθεί για να υποστηρίζουν machine – to – machine (M2M) δίκτυα και τις εφαρμογές τους.

Με την ζήτηση που απαιτείται για την εξυπηρέτηση ενός φάσματος IoT υπηρεσιών και εφαρμογών, όπως συναλλακτικές υπηρεσίες υψηλής ταχύτητας, context – aware εφαρμογές κ.α., πολλαπλά δίκτυα με διάφορες τεχνολογίες και πρωτόκολλα πρόσβασης χρειάζονται να δουλέψουν μεταξύ τους σε μία ετερογενή διαμόρφωση. Αυτά τα δίκτυα μπορούν να έχουν τη μορφή ενός ιδιωτικού, δημόσιου ή υβριδικού μοντέλου και είναι φτιαγμένα για να υποστηρίζουν τις απαιτήσεις επικοινωνίας για χρόνο καθυστέρησης, εύρους ζώνης ή ασφάλειας.

2.3.3 Management Service Layer:

Η υπηρεσία διαχείρισης καθιστά την επεξεργασία των πληροφοριών δυνατή μέσω των analytics, ελέγχους ασφάλειας, επεξεργασία μοντέλων και την διαχείριση των συσκευών.

Ένα από τα σημαντικά χαρακτηριστικά του management service layer είναι οι business and process μηχανές κανόνων. Το IoT φέρνει μαζί την σύνδεση και την αλληλεπίδραση των αντικειμένων με τα συστήματα, παρέχοντας έτσι πληροφορίες στην μορφή γεγονότων ή contextual data όπως η θερμοκρασία προϊόντων, τρέχουσα τοποθεσία και δεδομένα κυκλοφορίας. Μερικά από αυτά τα γεγονότα απαιτούν φιλτράρισμα ή δρομολόγηση σε post – processing συστήματα όπως την σύλληψη περιοδικών αισθητήριων δεδομένων, ενώ άλλα απαιτούν απόκριση σε πιο άμεσες καταστάσεις όπως την αντίδραση σε επείγουσες καταστάσεις στις συνθήκες υγείας ενός ασθενή. Οι μηχανές κανόνων υποστηρίζουν την τυποποίηση των λογικών απόφασης και ενεργοποιούν διαδραστικές και αυτόματες διεργασίες για να ενεργοποιήσουν ένα πιο αποκριτικό IoT σύστημα.

Στο πεδίο της ανάλυσης στοιχείων, διάφορα εργαλεία ανάλυσης χρησιμοποιούνται για να εξάγουν πληροφορίες από τεράστια ποσά «ωμών» δεδομένων και να επεξεργαστούν σε ένα πιο γρήγορο ρυθμό. Η ανάλυση στοιχείων όπως τα in – memory analytics επιτρέπουν μεγάλους όγκους δεδομένων να αποθηκευτούν προσωρινά στην μνήμη τυχαίας προσπέλασης (RAM) παρά να αποθηκευτούν σε φυσικούς δίσκους. Τα in – memory analytics μειώνουν τον χρόνο data query και ενισχύουν την ταχύτητα λήψης αποφάσεων. Τα streaming analytics είναι άλλη μία μορφή ανάλυσης στοιχείων όπου η ανάλυση δεδομένων, θεωρούμε σαν data – in – motion, απαιτείται να πραγματοποιείται σε πραγματικό χρόνο έτσι ώστε να μπορεί να γίνει λήψη αποφάσεων μέσα σε δευτερόλεπτα.

Η διαχείριση δεδομένων είναι η ικανότητα του να διαχειρίζεσαι την ροή δεδομένων πληροφορίας. Με την διαχείριση δεδομένων στο management service layer, οι πληροφορίες μπορούν να είναι προσβάσιμες, να ενσωματωθούν και να ελεγχθούν. Εφαρμογές υψηλότερου επιπέδου μπορούν να προστατευθούν από την ανάγκη για επεξεργασία περιττών δεδομένων και να μειωθεί το ρίσκο της αποκάλυψης απορρήτου. Τεχνικές φιλτραρίσματος δεδομένων όπως η ανωνυμία δεδομένων, η ενσωμάτωση δεδομένων και ο συγχρονισμός δεδομένων, χρησιμοποιούνται για να κρυφτούν οι λεπτομέρειες των πληροφοριών ενώ παρέχουν μόνο ουσιώδης πληροφορίες που είναι χρήσιμες για τις σχετικές εφαρμογές. Με την χρήση της άντλησης δεδομένων πληροφορίες μπορούν να εξαχθούν για να παρέχουν μία κοινή επιχειρηματική άποψη των δεδομένων προκειμένου να αποκτηθεί μία μεγαλύτερη ευελιξία και επαναχρησιμοποίηση σε όλους τους τομείς.

Η ασφάλεια πρέπει να επιβληθεί σε όλες τις διαστάσεις της IoT αρχιτεκτονικής, από το έξυπνο αντικείμενο μέχρι και το επίπεδο εφαρμογής (Application Layer). Η ασφάλεια του συστήματος προλαμβάνει hacking του συστήματος και εκθέσεις από μη – εξουσιοδοτημένα άτομα, οπότε μειώνει την πιθανότητα ρίσκου.

2.3.4 Application Layer:

Αυτό το επίπεδο παρέχει «παγκόσμια» διαχείριση της εφαρμογής βασιζόμενο στις πληροφορίες των αντικειμένων. Με άλλα λόγια ο χρήστης διαδρά με το δίκτυο των αισθητήρων μέσω αυτού του επιπέδου. Η IoT εφαρμογή καλύπτει «έξυπνα» περιβάλλοντα/χώρους σε τομείς όπως τα μέσα μαζικής μεταφοράς, κτήρια, πόλεις, lifestyle, λιανική πώληση, γεωργίας, εργοστασίων, αλυσίδας προμηθειών, επείγοντα, υγείας, διάδραση με τον χρήστη, πολιτισμού και τουρισμού, περιβάλλοντος και ενέργειας κ.α.

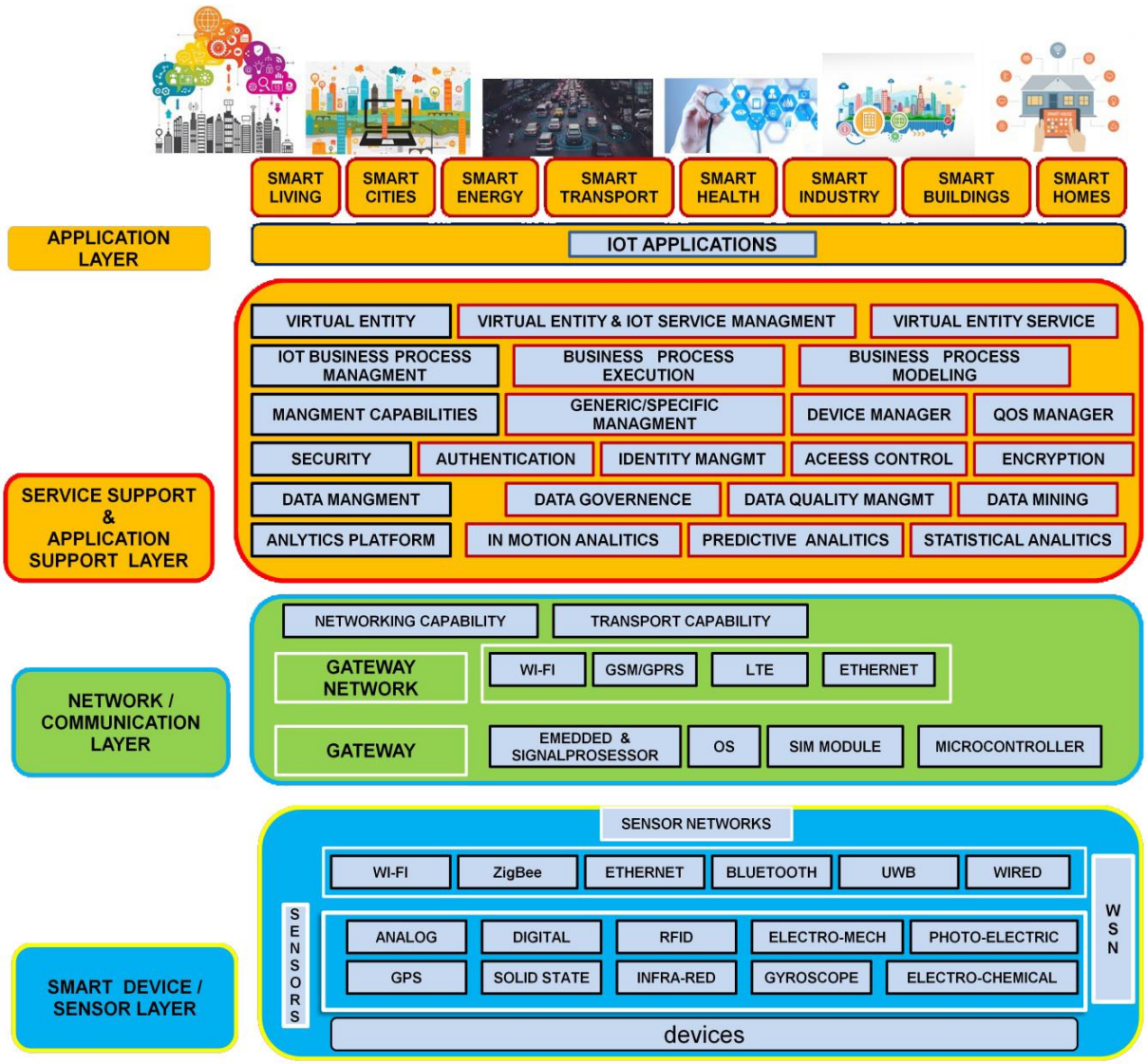

Figure 2-3 IoT architecture layers

2.4 Τεχνολογίες

Υπάρχουν πολλές τεχνολογίες που καθιστούν το Internet of Things. Κρίσιμο στο πεδίο είναι το δίκτυο που χρησιμοποιείται για την επικοινωνία μεταξύ των συσκευών μίας Internet of Things εγκατάστασης, ένα ρόλο που αρκετές ασύρματες ή ενσύρματες τεχνολογίες μπορούν να εκπληρώσουν.

2.4.1 Διευθυνσιοδότηση

Η αρχική ιδέα του Auto – ID Center είναι βασισμένη σε RFID – tags και την μοναδική ταυτοποίηση μέσω του ηλεκτρονικού κωδικού προϊόντος. Ωστόσο. Αυτό έχει εξελιχθεί με τα αντικείμενα να έχουν IP διεύθυνση ή Uniform Resource Identifier (URI). Μία εναλλακτική ματιά, από τον κόσμο του σημασιολογικού ιστού επικεντρώνεται στο να μην κάνει όλα τα αντικείμενα (όχι μόνο τα ηλεκτρονικά, smart ή RFID – enabled) addressable από τα υπάρχοντα πρωτόκολλα ονομασίας, όπως το URI. Τα αντικείμενα από μόνα τους δεν αντιστρέφονται, αλλά τώρα μπορούν να αναφερθούν από άλλους αντιπροσώπους όπως δυνατούς συγκεντρωμένους servers που δρουν για τον άνθρωπο – ιδιοκτήτη τους.

Η ενσωμάτωση με το Internet υποδηλώνει ότι οι συσκευές θα χρησιμοποιούν μία διεύθυνση IP σαν ένα μοναδικό αναγνωριστικό. Λόγω του περιορισμένου χώρου IPv4 διευθύνσεων (που επιτρέπει 4,3 δισεκατομμύρια μοναδικές διευθύνσεις), τα αντικείμενα στο IoT θα πρέπει να χρησιμοποιήσουν το νέας γενιάς πρωτόκολλο Internet (IPv6) για την κλιμάκωση στον εξαιρετικά μεγάλο αριθμό διευθύνσεων που απαιτείται. Οι internet – of – things συσκευές θα ωφεληθούν από το ανιθαγεννές παρόν αυτοδιαμόρφωσης διεύθυνσης στο IPv6, καθώς μειώνει την επιβάρυνση διαμόρφωσης τους κεντρικούς υπολογιστές και την IEFT 6LoWPAN συμπίεση. Σε ένα μεγάλο βαθμό το μέλλον του Internet of Things δεν θα είναι δυνατόν χωρίς την υποστήριξη του IPv6 και συνεπώς, η παγκόσμια υιοθεσία του IPv6 στα επόμενα χρόνια θα είναι κρίσιμη για την επιτυχή ανάπτυξη του IoT στο μέλλον.

2.4.2 Ασύρματες Τεχνολογίες Μικρής Εμβέλειας

- **Bluetooth mesh networking:** Προδιαγραφή που παρέχει μία παραλλαγή πλέγματος δικτύου σε Bluetooth Low Energy (BLE) με αυξημένο αριθμό κόμβων και τυποποιημένο επίπεδο εφαρμογής (μοντέλα).
- **Light – Fidelity (Li – Fi):** Τεχνολογία ασύρματης επικοινωνίας παρόμοια με το πρότυπο του Wi – Fi, αλλά χρησιμοποιεί επικοινωνία με ορατό φως για αυξημένο εύρος ζώνης.
- **Near Field Communication (NFC):** Πρωτόκολλα επικοινωνίας που επιτρέπουν 2δύο ηλεκτρονικές συσκευές να επικοινωνούν μέσα σε μία εμβέλεια τεσσάρων εκατοστών.
- **QR codes και Barcodes:** Οπτικές ετικέτες για ανάγνωση από μηχανή που αποθηκεύουν πληροφορία που αφορά το αντικείμενο το οποίο είναι συνημμένο.
- **Radio Frequency Identification (RFID):** Τεχνολογία που χρησιμοποιεί ηλεκτρομαγνητικά πεδία για να διαβάζει δεδομένα που είναι αποθηκευμένα σε ενσωματωμένες ετικέτες σε άλλα αντικείμενα.
- **Thread:** Δικτυακό πρωτόκολλο βασισμένο στο IEEE 802.15.4 πρότυπο, παρόμοιο με το ZigBee, που παρέχει IPv6 διευθυνσιοδότηση.
- **Transport Layer Security:** Δικτυακό πρωτόκολλο ασφαλείας.
- **Wi – Fi (Wireless Fidelity):** Χρησιμοποιείται ευρέως σαν τεχνολογία για τοπική δικτύωση. Είναι βασισμένο στο IEEE 802.11 πρότυπο, όπου οι συσκευές μπορούν να επικοινωνήσουν μέσω ενός κοινού σημείου πρόσβασης.
- **Wi – Fi Direct:** Αυτό αποτελεί παραλλαγή του Wi – Fi προτύπου για peer – to – peer επικοινωνία, εξαλείφοντας την ανάγκη για ένα σημείο πρόσβασης.
- **Z – Wave:** Πρωτόκολλο επικοινωνίας που παρέχει μικρής εμβέλειας, χαμηλής καθυστέρησης μεταφορά δεδομένων σε ρυθμούς και κατανάλωση ενέργειας μικρότερα από το Wi – Fi. Χρησιμοποιείται κυρίως σε οικιακούς αυτοματισμούς.
- **ZigBee:** Πρωτόκολλα επικοινωνίας για personal area networking βασισμένα στο IEEE 802.15.4 πρότυπο, που παρέχει χαμηλή κατανάλωση ενέργειας, χαμηλό ρυθμό δεδομένων και υψηλό χρόνο διέλευσης.

2.4.3 Ασύρματες Τεχνολογίες Μέσης Εμβέλειας

- **HaLow:** Παραλλαγή του Wi – Fi προτύπου που παρέχει εκτεταμένο εύρος για χαμηλής κατανάλωσης ενέργειας επικοινωνία σε χαμηλότερο ρυθμό δεδομένων.
- **LTE – Advanced:** Προδιαγραφή υψηλής ταχύτητας επικοινωνίας για κινητά δίκτυα. Παρέχει ενισχύσεις στο LTE πρότυπο με εκτεταμένη κάλυψη, υψηλότερο χρόνο διέλευσης και χαμηλότερη καθυστέρηση.

2.4.4 Ασύρματες Τεχνολογίες Μεγάλης Εμβέλειας

- **Low – Power Wide – Area Networking (LPWAN):** Ασύρματα δίκτυα σχεδιασμένα να επιτρέπουν μεγάλης εμβέλειας επικοινωνία με ένα χαμηλό ρυθμό δεδομένων, μειώνοντας το κόστος ενέργειας και εκπομπής. Διαθέσιμες τεχνολογίες και πρωτόκολλα LPWAN είναι: LoRaWan, Sigfox, NB – IoT, Weightless.
- **Very Small Aperture Terminal (VSAT):** τεχνολογία δορυφορικής επικοινωνίας που χρησιμοποιεί μικρές κεραίες – πιάτα για στενής και ευρείας ζώνης δεδομένα.
- **Long – Range Wi – Fi Connectivity.**

2.4.5 Ενσύρματες Τεχνολογίες

- **Ethernet:** Γενικού σκοπού δικτυακό πρότυπο που χρησιμοποιεί συνεστραμμένο ζευγάρι καλωδίων και οπτικές ίνες για σύζευξη με hubs ή switches.
- **Multimedia over Coax Alliance (MoCA):** Προδιαγραφή που επιτρέπει διανομή υψηλής ανάλυσης βίντεο και περιεχομένου σε ολόκληρο το σπίτι μέσω υπάρχουσας ομοαξονικής καλωδίωσης.
- **Power – Line Communication (PLC):** Τεχνολογία επικοινωνίας που χρησιμοποιεί την ηλεκτρική καλωδίωση για τη μεταφορά ενέργειας και δεδομένων. Προδιαγραφές όπως το HomePlug χρησιμοποιούν PLC για την δικτύωση των IoT συσκευών.

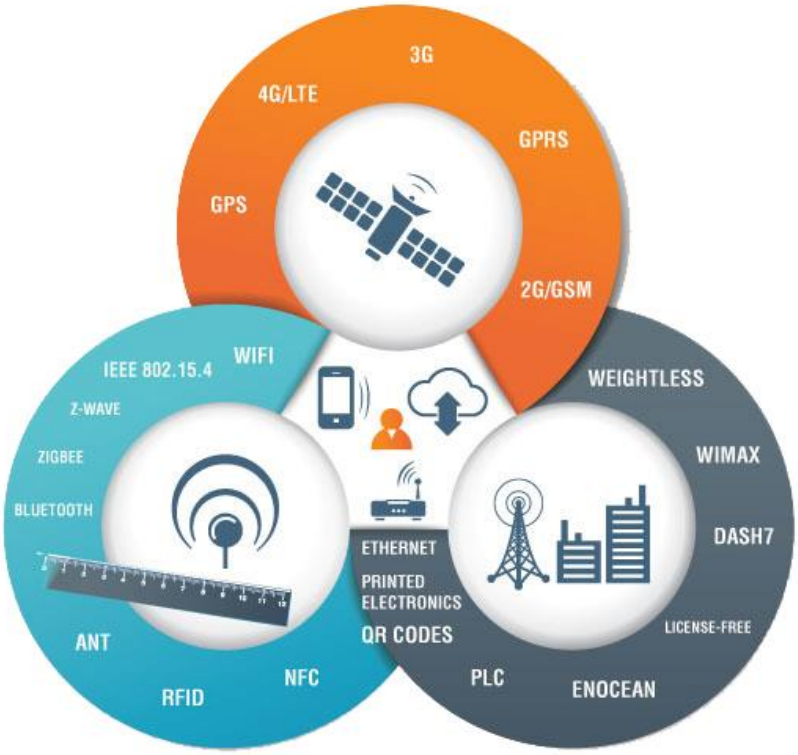

Figure 2-4 IoT Technologies

2.5 Πρωτόκολλα

2.5.1 Εισαγωγή

Όλο και περισσότερα πρωτόκολλα προστίθενται στο Internet of Things καθώς μεγάλοι προμηθευτές διευθετούν με αυτά τις ελλείψεις στα προϊόντα τους. Αυτά τα υψηλότερου επιπέδου IoT πρωτόκολλα είναι κατάλληλα για ένα ευρές φάσμα εφαρμογών. Για παράδειγμα, το MQTT έχει χρησιμοποιηθεί για πολλά χρόνια για την διαχείρηση των μυνημάτων μεταξύ server applications και έχει επικαιροποιηθεί για να διευθετεί την χρήση μικρού ασφαλούς client. Το DDNS έχει χρησιμοποιηθεί για να παρέχει πρόσβαση σε browser σε web συσκευές και το CoAP έχει επεκταθεί με άλλα πρωτόκολλα για να παρέχει διαχείριση ασφάλειας και εύρωστη λειτουργεία.

Το να γνωρίζουμε το σωστό πρωτόκολλο ή σετ πρωτοκόλλων για μια δοθείσα εφαρμογή το οποίο καλύπτει την επικοινωνία, την ασφάλεια, την διαχείριση και την κλιμάκωση είναι το πρώτο ζήτημα σχεδιασμού. Μετά από αυτό, η καλύτερη εφαρμογή του κάθε πρωτοκόλλου πρέπει να γίνει κατανοητή. Από αυτή την κατανόηση, ο σχεδιαστής μπορεί να διαλέξει την άριστη εφαρμογή του κάθε πρωτοκόλλου για το σύστημα και μετά από αυτά να διαλέξει το σωστό σετ για την εφαρμογή. Αυτή η απόφαση θα επηρεαστεί από αποφάσεις για τις απαιτήσεις σχετικά με το hardware που έχει επιλεγεί.

Στον χώρο των IoT πρωτοκόλλων, τα πρότυπα δεν συγκλίνουν ακόμα για συγκεκριμένες εφαρμογές και η αγορά τελικά αποφασίζει πιο από τα πρότυπα είναι πιο σχετικό. Αυτό αποτελεί πρόβλημα και ευκαιρία. Το πρωτόκολλο που θα επιλεγεί σήμερα μπορεί να θεωρείται απαρχαιωμένο στο μέλλον και να χρειάζεται να αντικατασταθεί. Αντιστρόφως, θα μπορούσε να γίνει το πιο δημοφιλές πρότυπο στο μέλλον.

2.5.2 IoT ή M2M Πρωτόκολλα

Υπάρχει ένα ευρύ σετ πρωτοκόλλων τα οποία προωθούνται σαν την ραχοκοκαλιά της IoT επικοινωνίας για το υψηλότερο επίπεδο machine – to – machine (M2M) πρωτόκολλο στην στοίβα των πρωτοκόλλων. Αυτά τα IoT ή M2M πρωτόκολλα εστιάζουν στην μεταφορά και επεξεργασία δεδομένων της εφαρμογής αν και μερικά όπως το SNMP είναι εστιασμένα περισσότερο στην απομακρυσμένη διαχείριση κόμβου. Η ακόλουθη λίστα συνοψίζει τα πρωτόκολλα που γενικά χρησιμοποιούνται. Στον πίνακα 1 παρακάτω γίνεται και η σύγκριση μεταξύ τους όσων αφορά τον τρόπο επικοινωνίας, τις απαιτήσεις συστήματος, τον τρόπο μεταφοράς και άλλων χαρακτηριστικών που τα διέπουν.

- **Azure – IoT**
- **CoAP**
- **Continua – Home Health Devices**
- **DDS**
- **DPWS: WS – Discovery, SOAP, WSAdressing, WDSL & XML Schema**
- **HTTP/REST**
- **MQTT & MQTT – SN/S**
- **SNMP**
- **Thread**
- **UPnP**
- **XMPP**
- **ZeroMQ**

Πρωτόκολλο	Μεταφορά	Messaging	2G, 3G, 4G	Low Power and Lossy	Compute Resources	Security	Success Stories	Arch
Azure – IoT	AMPQ/HTTP/TCP	Rqst/Rspnse	Excellent	Good	10K–100Ks RAM Flash	High Mandatory	Wearables	Client-Server
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium Optional	Utility field area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/ RAM Flash	High Optional	Military, Industrial	Bus
DPWS	TCP	Pub/Subsrb Rqst/Rspnse	Good	Fair	100Ks/ RAM Flash	High Optional	Web Servers	Client-Server
HTTP/REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low Optional	Smart Energy Phase 2	Client-Sernver
MQTT & MQTT- SN/S	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High Optional	Network Monitoring	Client-Server
Thread	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	High Mandatory	Nest?	Mesh
UPnP	UDP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client-Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High Mandatory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High Optional	Cern	P2P

Table 1 Χαρακτηριστικά Πρωτοκόλλων

2.5.3 Χαρακτηριστικά – Κλειδία των Πρωτοκόλλων

Οι επικοινωνία στο Internet of Things είναι βασισμένη στα Internet TCP/UDP πρωτόκολλα και τα συσχετισμένα πρωτόκολλα Internet για οργάνωση, πράγμα που σημαίνει οποιοδήποτε UDP datagram για TCP stream sockets. Οι Προγραμματιστές μικρών συσκευών υποστηρίζουν πως το UDP προσφέρει μεγάλα πλεονεκτήματα στην απόδοση και στο μέγεθος το οποίο θα ελαχιστοποιήσει το κόστος.

Τα stream sockets υποφέρουν στην απόδοση αλλά εγγυούνται σωστή παράδοση των δεδομένων χωρίς σφάλματα. Η απώλεια στην απόδοση για την μεταφορά δεδομένων αισθητήρα σε ένα STM32F4 στα 167MHz, για παράδειγμα, είναι λιγότερη από 16,7% (μετρημένη με 2KB πακέτα, μικρότερα πακέτα μειώνουν την απώλεια απόδοσης). Χρησιμοποιώντας stream sockets, μπορούμε επίσης να χρησιμοποιήσουμε standard security πρωτόκολλα. Παρομοίως, η διαφορά στο κόστος μνήμης για επιπλέον 20K flash και 8K RAM για την αναβάθμιση σε TCP είναι γενικά μικρό.

Είναι πολύ σημαντική η προσέγγιση της ανταλλαγής μηνυμάτων στο IoT και πολλά πρωτόκολλα έχουν γυρίσει σε ένα publish subscribe μοντέλο. Με πολλούς κόμβους να συνδέονται και να αποσυνδέονται, και αυτοί οι κόμβοι να πρέπει να συνδεθούν σε μια ποικιλία εφαρμογών στο cloud, το publish/subscribe request/response μοντέλο έχει ένα πλεονέκτημα. Αντιδρά δυναμικά σε τυχαίες on/off λειτουργείες και μπορεί να υποστηρίξει πολλούς κόμβους.

Δύο πρωτόκολλα, το CoAP και HTTP/REST είναι και τα δύο βασισμένα σε request/response χωρίς publish/subscribe προσέγγιση. Στην περίπτωση του CoAP η χρήση του

6loWPAN και η αυτόματη διευθυνσιοδότηση για IPv6 χρησιμοποιείται για να αναγνωρίζει μοναδικούς κόμβους. Στην περίπτωση του HTTP/REST η προσέγγιση είναι διαφορετική καθώς το αίτημα μπορεί να είναι οτιδήποτε, συμπεριλαμβάνοντας ένα αίτημά για publish ή ένα αίτημα για subscribe. Αυτά τα πρωτόκολλα συγχωνεύονται για την παροχή ενός πλήρους publish/subscribe request/response μοντέλου με το Thread σαν παράδειγμα.

Οι αρχιτεκτονικές συστήματος διαφέρουν, συμπεριλαμβάνοντας client – server, tree ή star, bus, και P2P. Η πλειοψηφία χρησιμοποιεί client – server αλλά άλλες χρησιμοποιούν bus και P2P προσεγγίσεις. Προβλήματα απόδοσης υπάρχουν για αυτές τις αρχιτεκτονικές με την καλύτερη απόδοση γενικότερα να εντοπίζεται στις P2P και bus αρχιτεκτονικές. Προσεγγίσεις από προσομοίωση ή από πρωτότυπα, προτιμώνται στο αρχικό στάδιο σχεδιασμού για προστασία εναντίον απρόσμενων γεγονότων.

Η κλιμάκωση εξαρτάται από την προσθήκη πολλών κόμβων στο σύστημα και τη δυνατότητα να αυξάνουμε τους πόρους εύκολα στο cloud για την εξυπηρέτηση αυτών των νέων κόμβων. Οι διάφορες αρχιτεκτονικές έχουν διάφορες ιδιότητες. Για client server αρχιτεκτονική, η αύξηση των διαθέσιμων servers είναι αποδοτική και εύκολη. Για bus και P2P αρχιτεκτονικές, η κλίμακα είναι έμφυτη στην αρχιτεκτονική αλλά δεν υπάρχουν υπηρεσίες cloud. Στην περίπτωση της tree ή star συνδεόμενης αρχιτεκτονικής, μπορούν να υπάρχουν προβλήματα σχετικά με την προσθήκη περισσότερων φύλλων στο δέντρο, πράγμα που σημαίνει πως επιβαρύνονται οι κόμβοι επικοινωνίας.

Μία άλλη όψη της κλιμάκωσης είναι η αντιμετώπιση ενός μεγάλου αριθμού μεταβαλλόμενων κόμβων και η σύνδεσή τους σε εφαρμογές cloud. Όπως προαναφέραμε τα publish/subscribe request/response συστήματα προορίζονται για κλιμάκωση επειδή αντιμετωπίζουν κόμβους που αποσυνδέονται για διάφορους λόγους, επιτρέπουν σε εφαρμογές να λαμβάνουν συγκεκριμένα δεδομένα όταν αποφασίζουν να κάνουν subscribe και request, με αποτέλεσμα την καλή ροή δεδομένων. Λιγότερο εύρωστες προσεγγίσεις δεν κλιμακώνονται τόσο καλά.

Δίκτυα απώλειας και χαμηλής κατανάλωσης (Low power and Lossy Networks) που ενεργοποιούνται και απενεργοποιούνται. Αυτή η δυναμική συμπεριφορά μπορεί να επηρεάσει ολόκληρους τομείς του δικτύου οπότε τα πρωτόκολλα σχεδιάζονται για δυναμική επαναδιαμόρφωση πολλαπλών «μονοπατιών». Συγκεκριμένα πρωτόκολλα δυναμικής δρομολόγησης που βρίσκονται στο ZigBee, ZigBee IP (χρησιμοποιώντας 6loWPAN) και ατόφιο 6loWPAN εξασφαλίζουν ότι το δίκτυο προσαρμόζεται. Χωρίς αυτά τα χαρακτηριστικά, η αντιμετώπιση αυτών των κόμβων γίνεται μία ασυνεχής λειτουργεία και κάνει τις απαιτήσεις πόρων αυτών των κόμβων αρκετά υψηλότερες.

Οι απαιτήσεις πόρων αποτελούν κλειδί καθώς ο όγκος της εφαρμογής αυξάνει. Οι μικροελεγκτές προσφέρουν ευφυία σε πολύ χαμηλό κόστος και έχουν την δυνατότητα να αντιμετωπίσουν τα προβλήματα που αναφέρονται παραπάνω. Μερικά πρωτόκολλα είναι πιο απαιτητικά στους πόρους για να είναι πρακτικά σε μικρούς κόμβους. Θα υπάρχουν περιορισμοί γύρω από την ασυνεχή λειτουργεία και την αποθήκευση δεδομένων, εκτός και αν συγκεκριμένα ποσά σειριακού flash ή άλλου αποθηκευτικού χώρου συμπεριλαμβάνονται. Καθώς οι πόροι αυξάνονται, για την μείωση του συνολικού κόστους του συστήματος κόμβοι συσσωμάτωσης είναι πιο πιθανό να προστεθούν για την παροχή επιπλέον κοινού αποθηκευτικού χώρου.

Η διαλειτουργικότητα θα είναι ουσιαστική για τις περισσότερες συσκευές στο μέλλον. Μέχρι στιγμής έχουμε δει σύνολα λύσεων αλλά αυτό που θέλουνε οι χρήστες είναι αισθητήρες και συσκευές να δουλεύουν μαζί. Χρησιμοποιώντας ένα σετ προτυποποιημένων πρωτοκόλλων όπως και προτυποποιημένης ανταλλαγής μηνυμάτων, οι συσκευές μπρούν να διαχωριστούν από τις υπηρεσίες cloud που τις υποστηρίζουν. Αυτή η προσέγγιση μπορεί να παρέχει ολοκληρωμένη διαλειτουργικότητα συσκευών. Επίσης χρησιμοποιώντας έξυπνές publish/subscribe επιλογές, διάφορες συσκευές μπορούν να χρησιμοποιήσουν ακόμα κα τις ίδιες υπηρεσίες cloud και να παρέχουν διάφορα χαρακτηριστικά. Χρησιμοποιώντας μία γενική

προσέγγιση, τα πρότυπα της εφαρμογής θα αναδυθούν, αλλά σήμερα, τα M2M πρότυπα μόλις αναδύονται, και τα πρότυπα εφαρμογής είναι αρκετά μακριά στο μέλλον. Όλα τα κύρια πρωτόκολλα προτυποποιούνται σήμερα.

Η χρήση standard λύσεων ασφάλειας της επιστήμης της πληροφορικής είναι ο πυρήνας των μηχανισμών ασφαλείας για τα περισσότερα πρωτόκολλα που προσφέρουν ασφάλεια. Αυτές οι προσεγγίσεις ασφάλειας είναι βασισμένες σε:

- TLS
- IPSec / VPN
- SSH
- SFTP
- Secure bootloader and automatic fallback
- Filtering
- HTTPS
- SNMP v3
- Encryption and decryption
- DTLS (για UDP ασφάλεια μόνο)

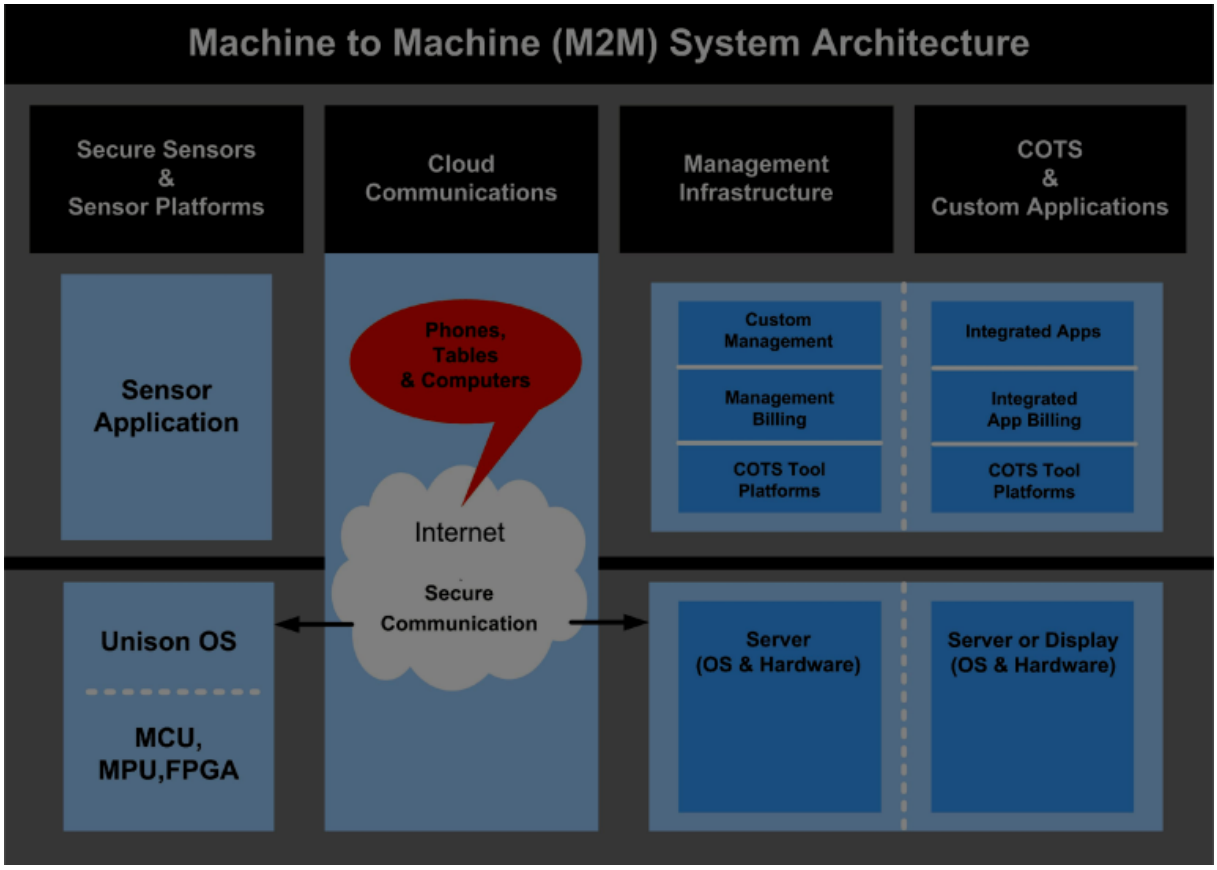

Figure 2-5 Σύστημα Αρχιτεκτονικής Machine to Machine (M2M)

2.5.4 MQTT

Figure 2-6 MQTT logo

Στην παρούσα εργασία η υλοποίηση του εργαστηριακού κομματιού έγινε με την χρήση του πρωτοκόλλου MQTT (Message Queuing Telemetry Transport). Πιο αναλυτικά, αυτό το πρωτόκολλο είναι ένα πρότυπο ISO (ISO / IEC 20922 PRF). Είναι ένα ελαφρύ πρωτόκολλο δημοσίευσης-εγγραφής (publish – subscribe) μηνυμάτων για χρήση πάνω από το πρωτόκολλο TCP / IP. Είναι σχεδιασμένο για συνδέσεις με απομακρυσμένες τοποθεσίες όπου απαιτείται ένα "μικρό ίχνος κώδικα" ή το εύρος ζώνης του δικτύου είναι περιορισμένο. Το πρότυπο απαιτεί έναν μεσίτη (broker). Ο broker είναι υπεύθυνος για τη διανομή μηνυμάτων σε ενδιαφερόμενους πελάτες με βάση το θέμα του μηνύματος.

Η προδιαγραφή του δεν προσδιορίζει την έννοια του μικρού κώδικα ή την έννοια του περιορισμένου εύρους ζώνης του δικτύου. Έτσι, η διαθεσιμότητα του πρωτοκόλλου για χρήση εξαρτάται από το πλαίσιο. MQTT-SN είναι μια παραλλαγή του κύριου πρωτοκόλλου που αποσκοπεί στην ενσωμάτωση συσκευών σε μη-TCP / IP δίκτυα, όπως το ZigBee.

Εναλλακτικά πρωτόκολλα περιλαμβάνουν το Advanced Message Queuing πρωτόκολλο, το IETF περιορισμένης εφαρμογής πρωτόκολλο, το XMPP και Web Application Messaging Protocol (WAMP).

Το MQTT ορίζει μεθόδους για να υποδεικνύουν την επιθυμητή δράση που πρέπει να εκτελεστεί στην ταυτοποιημένη πηγή. Τι αντιπροσωπεύει αυτή η πηγή, είτε προϋπήρχαν δεδομένα ή δεδομένα που δημιουργούνται δυναμικά, εξαρτάται από την υλοποίηση του server. Συχνά η πηγή αντιστοιχεί σε ένα αρχείο ή την έξοδο ενός εκτελέσιμου που φιλοξενούνται στον server.

Οι εντολές τις οποίες χρησιμοποιεί το MQTT για την μετάδοση των μηνυμάτων μεταξύ των συσκευών είναι οι εξής:

- **Connect**
  Περιμένει για μία σύνδεση που θα συσταθεί με τον server.
- **Disconnect**
  Περιμένει τον MQTT Client να τελειώσει να τελειώσει κάθε εργασία που έχει να κάνει, και για αποσύνδεση της συνεδρίας του TCP/IP.
- **Subscribe**
  Περιμένει την ολοκλήρωση μεθόδου εγγραφής ή διαγραφής.
- **UnSubscribe**
  Ζητά από τον server την διαγραφή του client από ένα η περισσότερα topics.
- **Publish**
  Μόλις κάνει την δημοσίευση στον MQTT Client επιστρέφει αμέσως στο νήμα της εφαρμογής.

Quality of Service (QoS)

Η ποιότητα υπηρεσιών (quality of service) αναφέρεται στους μηχανισμούς ελέγχου στην προτεραιότητα της κίνησης και την δέσμευση των πόρων παρά την επιτευχθείσα ποιότητα της υπηρεσίας. Η ποιότητα των υπηρεσιών είναι η ικανότητα να παρέχει διαφορετική προτεραιότητα σε διάφορες εφαρμογές, χρήστες ή ροές δεδομένων ή να εγγυείται σε ένα βαθμό την απόδοση της ροής δεδομένων.

Το MQTT πρωτόκολλο διαθέτει διάφορες βιβλιοθήκες για τον προγραμματισμό των συσκευών του συστήματός μας και η επιλογή της κατάλληλης έχει να κάνει καθαρά με την εφαρμογή που υλοποιούμαι και το hardware που χρησιμοποιούμε. Ο πίνακας παρακάτω μας τις συγκρίνει μεταξύ τους.

Library	Developed by	Language	Type	Release Date	Last release	Last release date	License
Adafruit IO	Adafruit	Ruby on Rails, Node.js	Client	?	2.0.0	?	?
M2Mqtt	eclipse	C#	Client	2017-05-20	4.3.0.0	2017-05-20	Eclipse Public License 1.0
Machine Head	ClojureWerkz Team	Clojure	Client	2013-11-03	1.0.0	2017-03-05	Creative Commons Attribution 3.0 Unported License
mosquette	Selva, Andrea	Java	Broker	2015-07-08	0.10	2017-06-30	Apache License 2.0
Mosquitto	eclipse	C, Python	Broker	2014-11-10	1.14.14	2017-07-11	Eclipse Public License 1.0, Eclipse Distribution License 1.0 (BSD)
Paho MQTT	eclipse	C, C++, Java, Javascript, Python, Go	Client	2014-05-02	1.4.0	2018-06-27	Eclipse Public License 1.0, Eclipse Distribution License 1.0 (BSD)
wofMQTT	wolfSSL	C	Client	2015-11-06	0.14	2017-11-22	GNU Public License, version 2

Table 2 MQTT Libraries

2.6 Ασύρματα Δίκτυα Αισθητήρων (WSN)

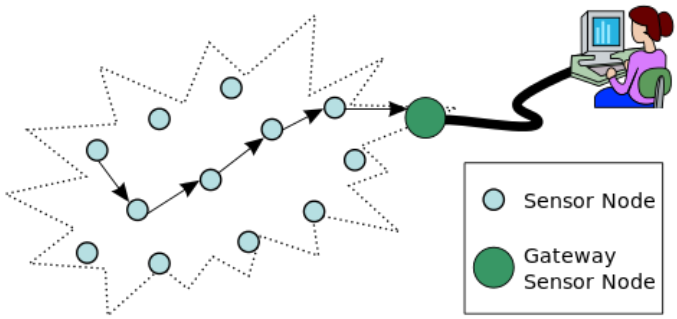

Figure 2-7 Wireless Sensor Network

2.6.1 Σφαιρική εικόνα

Ένα δίκτυο αισθητήρων αποτελείται από διασκορπισμένους αυτόνομους αισθητήρες για την παρακολούθηση φυσικών ή περιβαλλοντολογικών συνθηκών, όπως η θερμοκρασία, ο ήχος, η ατμοσφαιρική πίεση κτλ. Και μέσω συνεργασίας να μεταφέρει τα δεδομένα μέσω του δικτύου σε μια συγκεκριμένη τοποθεσία. Τα πιο μοντέρνα δίκτυα είναι ικανά και να δίνουν αλλά και να δέχονται πληροφορίες, πράγμα που τους επιτρέπει να ελέγχουν την δραστηριότητα των αισθητήρων.

Το κίνητρο για την ανάπτυξη των ασύρματων δικτύων με αισθητήρες ήταν οι στρατιωτικές εφαρμογές όπως η παρακολούθηση των πεδίων μάχης. Σήμερα τέτοια δίκτυα χρησιμοποιούνται σε πολλές καταναλωτικές και βιομηχανικές εφαρμογές, η παρακολούθηση και ο έλεγχος της βιομηχανικής παραγωγής, την παρακολούθηση των μηχανημάτων υγείας και πολλά άλλα.

Το ασύρματο δίκτυο αισθητήρων αποτελείται από κόμβους - από μερικές σε αρκετές εκατοντάδες η ακόμα και χιλιάδες, όπου κάθε κόμβος συνδέετε σε έναν (η κάποιες φορές σε αρκετούς) αισθητήρες. Κάθε τέτοιος κόμβος του δικτύου αισθητήρων έχει χαρακτηριστικά μερικά κομμάτια: ένα ραδιοπομποδέκτη με μια εσωτερική κεραία ή μια σύνδεση με μια εξωτερική κεραία, ένα μικροελεγκτή, ένα ηλεκτρονικό κύκλωμα για τη διασύνδεση με τους αισθητήρες και μια πηγή ενέργειας, συνήθως μια μπαταρία η μια ενσωματωμένη μορφή συγκομιδής ενέργειας.

Ένας αισθητήριος κόμβος μπορεί να ποικίλει σε μέγεθος από εκείνο ενός κουτιού παπουτσιών μέχρι το μέγεθος ενός κόκκου σκόνης, αν και λειτουργικοί «κόκκοι» πραγματικά μικροσκοπικών διαστάσεων δεν έχουν ακόμα δημιουργηθεί. Το κόστος των αισθητήριων κόμβων ποικίλει, ξεκινώντας από μερικά και φτάνοντας σε εκατοντάδες δολάρια, αναλόγως την πολυπλοκότητα των μεμονωμένων αισθητήριων κόμβων. Οι περιορισμοί σε μέγεθος και κόστος έχουν ως αποτέλεσμα αντίστοιχους περιορισμούς σε πόρους όπως ενέργεια, μνήμη, υπολογιστική ταχύτητα και στο εύρος ζώνης των επικοινωνιών. Η τοπολογία των αισθητήρων μπορεί να διαφέρει από ένα δίκτυο τοπολογίας αστέρος σε ένα αναπτυγμένο ασύρματο δίκτυο πλέγματος multi-hop.

2.6.2 Ασύρματα δίκτυα αισθητήρων & IoT

Ένα ασύρματο δίκτυο αισθητήρων αποτελεί κομμάτι της τοπολογίας του Internet of Things. Ένα τέτοιο δίκτυο αποτελεί τα «μάτια» και τα «αυτιά» ενός Internet of Things συστήματος. Είναι η γέφυρα που συνδέει τον αναλογικό κόσμο με τον ψηφιακό και είναι υπεύθυνο να περνάει τα δεδομένα από τον αληθινό – αναλογικό κόσμο στο Internet· μπορούμε, δηλαδή, να πούμε πως το ασύρματο δίκτυο αισθητήρων εμπλέκεται με την επικοινωνία υλικού (hardware).

Το Internet of Things σε μία ευρεία έννοια μπορεί να χαρακτηριστεί ως ο εγκέφαλος, καθώς μπορεί να αποθηκεύσει και τα δεδομένα του αληθινού κόσμου (σε cloud υπηρεσίες ή βάσεις δεδομένων) και μπορεί να χρησιμοποιηθεί για να παρακολουθεί τις παραμέτρους του αληθινού κόσμου. Επίσης μπορεί να ερμηνεύει ουσιαστικά το περιβάλλον και ακόμα και να παίρνει αποφάσεις σύμφωνα με τα δεδομένα από τα αισθητήρια.

2.6.3 Χαρακτηριστικά

Προσανατολισμός στην Εφαρμογή (Application Specific): Τα Ασύρματα Δίκτυα Αισθητήρων σχεδιάζονται με βάση τις εξειδικευμένες ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Σπανίως ένα Δίκτυο το οποίο έχει υλοποιηθεί για μία συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί (χωρίς προσαρμογές) σε κάποια άλλη εφαρμογή. Για παράδειγμα, ένα δίκτυο από αισθητήρες που έχουν ενταχθεί στην κεντρική θέρμανση ενός κτιρίου ρυθμίζοντας τη μέγιστη επιμέρους θερμοκρασία των ορόφων, στηρίζεται σε εντελώς διαφορετικές αρχές σχεδιασμού από ένα στατικό δίκτυο αισθητήρων που μετρούν το ύψος της βροχόπτωσης μιας γεωγραφικής περιοχής.

Κλίμακα Μεγέθους: Τα Ασύρματα Δίκτυα Αισθητήρων μπορούν να είναι όσο εκτενή ή όσο περιορισμένα απαιτεί η εφαρμογή, αποτελούμενα από μερικές δεκάδες έως και αρκετές εκατοντάδες αισθητήρες. Τόσο η αρχιτεκτονική όσο και τα δικτυακά πρωτόκολλα θα πρέπει να είναι σε θέση να διαχειριστούν τέτοια μεγέθη.

Αυτορδιαμόρφωση και Ανοχή Σφαλμάτων: Τα δίκτυα λειτουργούν με αλγορίθμους ικανούς να ρυθμίσουν την τοπολογία τους τη στιγμή της εγκατάστασης, αλλά και να διαχειριστούν τυχόν μελλοντικές ανάγκες για μεταβολές σε αυτή: χρησιμοποιούν αλγορίθμους ικανούς να αντιμετωπίσουν το φαινόμενο της απώλειας αισθητήρων (π.χ. λόγω φθοράς ή και κλοπής) με τρόπο ώστε να μη διαταράσσεται η συνολική σταθερότητα λειτουργίας τους. Η συνολική κατάσταση του Δικτύου ελέγχεται ανά τακτά χρονικά διαστήματα.

Χρόνος Ζωής: Όταν πρόκειται για δίκτυα αποτελούμενα από αισθητήρες οι οποίοι τροφοδοτούνται από ηλεκτρικά στοιχεία (π.χ. μπαταρίες), ο χρόνος ζωής τους είναι άμεσα συνυφασμένος με το χρόνο ζωής των πηγών ενέργειας των αισθητήρων. Σημαντικό ρόλο στη διατήρηση της βιωσιμότητας ενός δικτύου για μεγάλο χρονικό διάστημα παίζει ο προσεκτικός σχεδιασμός γύρω από τη διαχείριση της ενέργειας η οποία απαιτείται για τη λειτουργία του κάθε αισθητήρα. Η σχέση ανάμεσα στην ποιότητα λειτουργίας και την απαιτούμενη τροφοδοσία των αισθητήρων είναι αντιστρόφως ανάλογη: ξοδεύοντας περισσότερη ενέργεια, επιτυγχάνουμε μεν καλύτερη απόδοση αλλά το δίκτυο έχει μικρότερο συνολικό χρόνο ζωής. Ακριβώς επειδή τα δίκτυα αυτά σχεδιάζονται με τρόπο ώστε να εξυπηρετήσουν συγκεκριμένες ανάγκες, ο τρόπος εξισορρόπησης της σχέσης ποιότητας-ενέργειας εξαρτάται από το είδος της εφαρμογής.

Αυτόνομη Λειτουργία και Προγραμματισμός: Κάθε ένας από τους αισθητήρες πρέπει να είναι σε θέση να λάβει αποφάσεις για τη διατήρηση της εύρυθμης λειτουργίας του δικτύου και χωρίς την παρέμβαση του χρήστη, (π.χ. αυξομειώνοντας τη συχνότητα δειγματοληψίας) αλλά και να δεχθεί επαναπρογραμματισμό (π.χ. σε περίπτωση που αλλάξει η στρατηγική χρήσης του δικτύου).

Απλότητα Σχεδιασμού: Με δεδομένο ότι οι αισθητήρες διαθέτουν περιορισμένους πόρους λειτουργίας, δεν υπάρχει η δυνατότητα να υποστηριχθεί ιδιαιτέρως υψηλή πολυπλοκότητα, ούτε σε επίπεδο λειτουργικού συστήματος ούτε σε επίπεδο αλγορίθμου λειτουργίας του δικτύου.

Ποιότητα Υπηρεσιών (Quality of Service): Σε τέτοια δίκτυα η έννοια της ποιότητας υπηρεσιών μπορεί να διαφέρει σημαντικά από τα συνήθη πρότυπα των Ad Hoc δικτύων. Για παράδειγμα, σε ένα Ασύρματο Δίκτυο Αισθητήρων, θα μπορούσε να μην παίζει τόσο σημαντικό ρόλο η ταχύτητα μετάδοσης των δεδομένων όσο η αξιόπιστη μετάδοση του συνόλου χωρίς απώλειες, αποφεύγοντας τις επανεκπομπές.

Για να καλυφθούν οι παραπάνω απαιτήσεις έχουν αναπτυχθεί νέοι τρόποι ασύρματης επικοινωνίας μεταξύ των δομικών στοιχείων (κόμβων) του δικτύου. Οι κυριότεροι από αυτούς είναι:

- **Επικοινωνία Πολλαπλών Βημάτων (Multi-Hop):** Οι ενδιάμεσοι κόμβοι του δικτύου λειτουργούν ως «προωθητές» μηνυμάτων. Ένα τέτοιο είδος επικοινωνίας είναι χρήσιμο όταν η απευθείας σύνδεση δύο κόμβων είναι ανέφικτη (ή κρίνεται ασύμφορη, λόγω του υψηλού κόστους της ενέργειας μετάδοσης).
- **Δεδομενο-κεντρική Επικοινωνία (Data-Centric):** Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, μπορούν να ομαδοποιηθούν από την Κεντρική Μονάδα Παρακολούθησης βάσει του φαινομένου παρακολούθησης (και όχι π.χ. βάσει της διεύθυνσης IP).
- **Συνάθροιση Δεδομένων (Data Aggregation):** Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, υπάρχει η δυνατότητα να μεταδώσουν προς την Κεντρική Μονάδα Παρακολούθησης πλεονάζοντα (ή επικαλυπτόμενα) στοιχεία. Η «περιττή» πληροφορία εξαλείφεται ώστε να εξοικονομηθεί ενέργεια και να βελτιωθεί ο συνολικός ρυθμός απόκρισης του δικτύου.

Τοπολογίες

Για τα δίκτυα ράδιο – επικοινωνιών, η δομή των ασυρμάτων δικτύων αισθητήρων περιλαμβάνει διάφορες τοπολογίες. Μερικές από αυτές είναι η τοπολογία αστέρας (star topology), η τοπολογία δέντρο (tree topology) και η τοπολογία πλέγμα (mesh topology)

- **Star Topology:** Αυτή η τοπολογία είναι μία τοπολογία επικοινωνίας, όπου κάθε κόμβος συνδέεται απευθείας σε ένα gateway. Ένα gateway μπορεί να στέλνει ή να λαμβάνει ένα μήνυμα σε έναν αριθμό από απομακρυσμένους κόμβους. Στις τοπολογίες αστέρα, οι κόμβοι δεν τους επιτρέπεται να στέλνουν μηνύματα ο ένας στον άλλο. Αυτό επιτρέπει low – latency επικοινωνία μεταξύ του κόμβου και του gateway (base station).

  Λόγω της εξάρτησης της τοπολογίας σε ένα κόμβο για την διαχείριση του δικτύου, gateway πρέπει να είναι μέσα στην εμβέλεια της ράδιο – μετάδοσης όλων των κόμβων. Πλεονέκτημα αποτελεί την ικανότητα να διατηρεί τους απομακρυσμένους κόμβους σε κατάσταση ελάχιστης κατανάλωσης ενέργειας και απλά υπό έλεγχο. Το μέγεθος του δικτύου εξαρτάται από τον αριθμό των συνδέσεων στο hub.
- **Tree Topology:** Αυτή η τοπολογία λέγεται επίσης και κατακερματισμένη τοπολογία αστέρα. Σε τοπολογίες δέντρων, κάθε κόμβος συνδέεται σε ένα κόμβο που είναι τοποθετημένος ψηλότερα στο δέντρο και κατόπιν στο gateway. Το κύριο πλεονέκτημα της τοπολογίας δέντρου είναι ότι η επέκταση ενός δικτύου είναι ευκόλως δυνατή και επιπλέον η εύρεση σφάλματος είναι εύκολη. Το μειονέκτημα σε αυτό το δίκτυο είναι πως βασίζεται πολύ στο καλώδιο διαύλου· αν αυτό σπάσει, όλο το δίκτυο θα καταρρεύσει.
- **Mesh Topology:** Σε αυτή την τοπολογία επιτρέπεται στους κόμβους να ανταλλάσσουν δεδομένα, εφ’ όσον ο άλλος κόμβος είναι μέσα στην εμβέλεια. Αν ένας κόμβος θέλει να στείλει ένα μήνυμα σε κάποιον κόμβο εκτός εμβέλειας θα χρειαστεί ένα 3^{ο} κόμβο μεσάζοντα για να προωθήσει το μήνυμα. Το πλεονέκτημα είναι ότι έχουμε εύκολη απομόνωση κόμβου και ευκολία στον εντοπισμό σφαλμάτων στο δίκτυο. Το μειονέκτημα είναι πως είναι μεγάλο και χρειάζεται μεγάλη επένδυση.

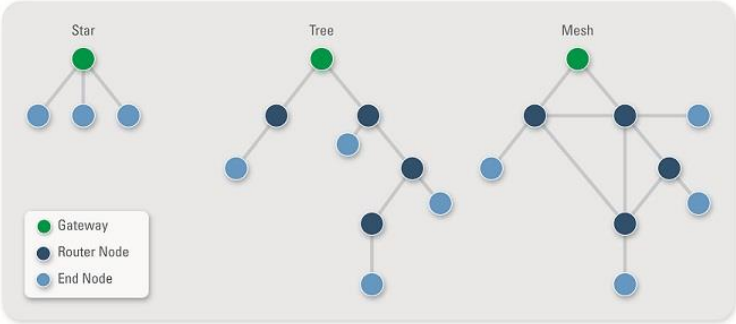

Figure 2-8 Wireless Sensor Network Topologies

Τύποι

Ανάλογα με το περιβάλλον, αποφασίζουμε και το τι τύπος δικτύου θα αναπτυχθεί. Διάφοροι τύποι δικτύου είναι οι εξής:

- **Επίγειο WSN:** Τα επίγεια ασύρματα δίκτυα αισθητήρων είναι ικανά για να επικοινωνούν με τα base stations αποδοτικά και αποτελούνται από εκατοντάδες ή και χιλιάδες ασύρματους κόμβους αισθητήρων παραταγμένους είτε σε μη – δομημένο ή δομημένο τρόπο. Σε μη – δομημένο τρόπο, οι κόμβοι των αισθητήρων διανέμονται τυχαία μέσα στην περιοχή που θέλουμε. Ο δομημένος τρόπος απαιτεί άριστη τοποθεσία, grid placement, και 2D/3D μοντέλα τοποθεσίας.

  Σε αυτό τον τύπο WSN η τροφοδοσία είναι περιορισμένη. Ωστόσο η μπαταρία είναι εξοπλισμένη με ηλιακά κύτταρα σαν δευτερεύουσα πηγή ενέργειας. Η διατήρηση της ενέργειας αυτών των WSN επιτυγχάνεται χρησιμοποιώντας low duty cycle operations, ελαχιστοποιώντας τις καθυστερήσεις, άριστη δρομολόγηση κ.α.
- **Υπόγειο WSN:** Τα υπόγεια ασύρματα δίκτυα αισθητήρων είναι πιο ακριβά από τα επίγεια όσον αφορά το κόστος ανάπτυξης, συντήρηση, το κόστος εξοπλισμού και την προσεχτική σχεδίαση. Το δίκτυο αποτελείται από ένα αριθμό κόμβων που είναι κρυμμένοι κάτω από το έδαφος για την παρακολούθηση υπόγειων καταστάσεων. Για την αναμετάδοση πληροφοριών από τους κόμβους των αισθητήρων στο base station, επιπλέον νεκροί κόμβοι τοποθετούνται πάνω από το έδαφος.

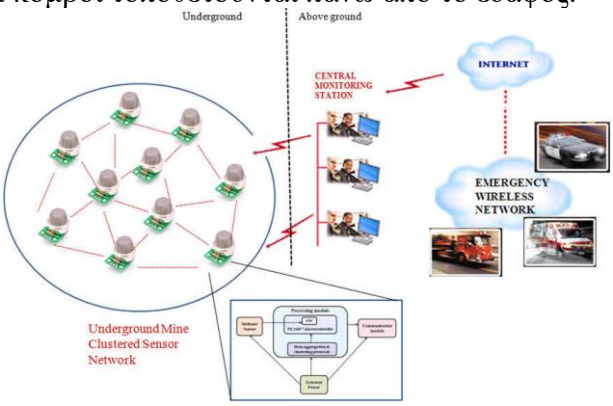

Figure 2-9 Υπόγειο Wireless Sensor Network

Τα υπόγεια ασύρματα δίκτυα αισθητήρων είναι δύσκολο να επαναφορτιστούν. Επιπλέον το υπόγειο περιβάλλον αποτελεί πρόκληση για την ασύρματη επικοινωνία λόγω της υψηλής απόσβεσης και απώλειας σήματος.

- **Υποθαλάσσιο WSN:** Αυτά τα δίκτυα αποτελούνται από ένα αριθμό αισθητήρων και οχημάτων που έχουν αναπτυχθεί κάτω από τη θάλασσα, Αυτόνομα υποθαλάσσια οχήματα χρησιμοποιούνται για τη συλλογή δεδομένων από τους κόμβους αισθητήρων. Μία πρόκληση της υποθαλάσσιας επικοινωνίας είναι η μεγάλης διάρκειας καθυστέρηση μετάδοσης, το εύρος ζώνης και οι βλάβες αισθητήρα.

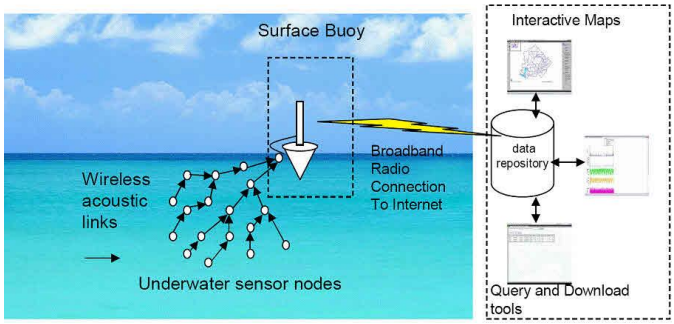

Figure 2-10 Υποθαλάσσιο Wireless Sensor Network

Και σε αυτόν το τρόπο έχουμε προβλήματα παροχής ενέργειας καθώς οι μπαταρίες δεν μπορούν να επαναφορτιστούν ή να αντικατασταθούν. Το πρόβλημα διατήρησης ενέργειας σε τέτοια δίκτυα συνεπάγει την ανάπτυξη της υποθαλάσσιας επικοινωνίας και τεχνικών δικτύωσης.

- **Multimedia WSN:** Αυτά τα δίκτυα έχουν προταθεί για να επιτραπεί η ανίχνευση και η παρακολούθηση γεγονότων σε πολυμεσική μορφή, όπως εικόνα, βίντεο και ήχο. Αυτά τα δίκτυα αποτελούνται από χαμηλού κόστους αισθητήρες με μικρόφωνα και κάμερες. Αυτοί οι κόμβοι είναι διασυνδεδεμένοι μεταξύ τους μέσω μίας ασύρματης σύνδεσης για την συμπίεση δεδομένων, την ανάκτησή τους και την συσχέτισή τους.

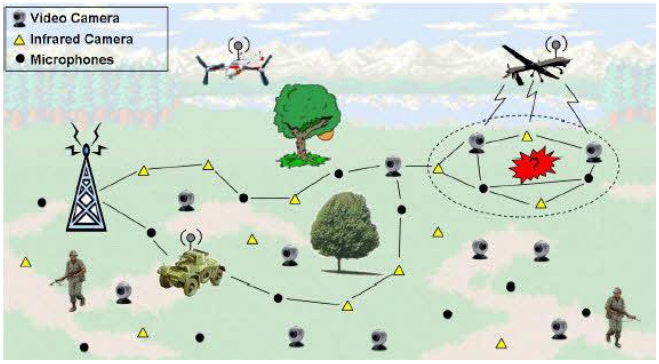

Figure 2-11 Multimedia Wireless Sensor Network

Η πρόκληση με αυτά τα δίκτυα είναι ότι έχουν μεγάλη κατανάλωση ενέργειας, απαιτήσεις υψηλού εύρους ζώνης και περιέχουν τεχνικές επεξεργασίας και συμπίεσης δεδομένων.

- **Κινητό WSN:** Αυτά τα δίκτυα αποτελούνται από μία συλλογή κόμβων αισθητήρων που μπορούν να μετακινηθούν από μόνοι τους και να διαδράσουν με το φυσικό περιβάλλον. Οι κινητοί κόμβοι έχουν τη δυνατότητα να επεξεργάζονται τα ερεθίσματα και να επικοινωνούν.

  Τα κινητά ασύρματα δίκτυα αισθητήρων είναι πολύ πιο πολύπλευρα από τα στατικά δίκτυα αισθητήρων. Τα πλεονεκτήματα των κινητών δικτύων σε σχέση με τα στατικά περιλαμβάνουν καλύτερη και βελτιωμένη εμβέλεια, καλύτερη ενεργειακή αποδοτικότητα, ανώτερη χωρητικότητα καναλιού κ.α.

2.6.4 Πλατφόρμες

- **Hardware**:

Μία μεγάλη πρόκληση σε ένα ασύρματο δίκτυο κόμβων αισθητήρων είναι η παραγωγή μικροσκοπικών και χαμηλού κόστους αισθητήρων. Υπάρχει ένας αύξων αριθμός μικρών εταιριών που παράγει WSN hardware και η εμπορική κατάσταση μπορεί να συγκριθεί με αυτή

της αγοράς των οικιακών υπολογιστών τη δεκαετία του 1970. Πολλοί από τους κόμβους είναι σε στάδιο έρευνας και ανάπτυξης, πρακτικά όσον αφορά το λογισμικό. Επίσης με την υιοθεσία του δικτύου αισθητήρων κληρονομείται και η χρήση των πολύ χαμηλής κατανάλωσης ενέργειας μεθόδων για ραδιοεπικοινωνία και απόκτηση δεδομένων.

Σε πολλές εφαρμογές, ένα ασύρματο δίκτυο αισθητήρων επικοινωνεί με ένα Local Area Network ή ένα Wide Area Network μέσω ενός gateway. Το gateway λειτουργεί σαν μια γέφυρα μεταξύ του ασύρματου δικτύου αισθητήρων και του άλλου δικτύου. Αυτό ενεργοποιεί την δυνατότητα για τα δεδομένα να αποθηκεύονται και να επεξεργάζονται από συσκευές με περισσότερους πόρους, για παράδειγμα, σε έναν απομακρυσμένο server. Ένα wireless wide area network που χρησιμοποιείται κυρίως για συσκευές χαμηλής κατανάλωσης ενέργειας είναι γνωστό και σαν Low – Power Wide – Area Network ή απλά LPWAN.

- **Wireless:**

Υπάρχουν αρκετά wireless standards και λύσεις για την συνδεσιμότητα των αισθητήρων. Το ZigBee μπορεί να συνδέει αισθητήρες που λειτουργούν στα 2,4 GHz με ρυθμό δεδομένων στα 250 Kbps. Το Z – Wave λειτουργεί στα 915 MHz και έχει μεγαλύτερο εύρος κάλυψης αλλά χαμηλότερο ρυθμό δεδομένων. Το IEEE 802.15.4 working group παρέχει ένα standard για συνδεσιμότητα συσκευών χαμηλής κατανάλωσης ενέργειας και συχνά αισθητήρες και smart meters χρησιμοποιούν ένα από αυτά τα standard για συνδεσιμότητα.

Με την εμφάνιση του Internet of Things, αρκετές άλλες προτάσεις έχουν γίνει για την παροχή συνδεσιμότητας στους αισθητήρες. Το LORA είναι μίας μορφής LPWAN το οποίο παρέχει μεγάλου εύρους και χαμηλής κατανάλωσης συνδεσιμότητα για συσκευές, οι οποίες έχουν χρησιμοποιηθεί σε smart meters. Το Wi – SUN συνδέει συσκευές στο σπίτι. Το NarrowBand IoT και το LTE – M μπορούν να συνδέσουν έως εκατομμύρια αισθητήρες και συσκευές χρησιμοποιώντας την κυτταρική τεχνολογία.

- **Software:**

Η ενέργεια είναι ο πιο «τρομακτικός» πόρος των κόμβων ασυρμάτων δικτύων αισθητήρων και καθορίζει τη διάρκεια ζωής του δικτύου. Τα ασύρματα δίκτυα αισθητήρων μπορούν να αναπτυχθούν σε μεγάλους αριθμούς σε διάφορα περιβάλλοντα, συμπεριλαμβανομένων των απόμακρων και δύσκολων περιοχών, όπου οι ad – hoc επικοινωνίες αποτελούν στοιχείο κλειδί. Για αυτό τον λόγο, αλγόριθμοι και πρωτόκολλα χρειάζονται για να διευθετηθούν τα ακόλουθα προβλήματα:

  - Αυξημένη διάρκεια ζωής.
  - Ευρωστία και ανοχή σφαλμάτων.
  - Αυτοδιαμόρφωση.

Μεγιστοποίηση διάρκειας ζωής: Η κατανάλωση ενέργειας της αισθητήριας συσκευής πρέπει να ελαχιστοποιηθεί και οι κόμβοι πρέπει να είναι ενεργειακά αυτάρκης αφού η περιορισμένη πηγή ενέργειας καθορίζει την διάρκεια ζωής τους. Για την διατήρηση ενέργειας, οι ασύρματοι κόμβοι αισθητήρων συνήθως κλείνουν τον ραδιοπομπό και τον ραδιοδέκτη όταν αυτός δε χρησιμοποιείται.

- **Λειτουργικά συστήματα:**

Τα λειτουργικά συστήματα για τους κόμβους ενός ασύρματου δικτύου αισθητήρων είναι τυπικά λιγότερο περίπλοκα από τα λειτουργικά συστήματα γενικής χρήσης. Πιο πολύ φέρνουν σε ενσωματωμένα συστήματα για δύο λόγους. Πρώτον, τα ασύρματα δίκτυα αισθητήρων συνήθως αναπτύσσονται για μία συγκεκριμένη εφαρμογή, παρά σαν μία πλατφόρμα γενικής χρήσης. Δεύτερον, η ανάγκη για χαμηλό κόστος και χαμηλή κατανάλωση ενέργειας οδηγεί τους περισσότερους ασύρματους κόμβους αισθητήρων να έχουν μικροελεγκτές χαμηλής κατανάλωσης ενέργειας, εξασφαλίζοντας ότι μηχανισμοί σαν την εικονική μνήμη είναι είτε περιττοί ή πολύ ακριβοί για να εφαρμοστούν.

Έτσι λοιπόν, είναι δυνατό να χρησιμοποιήσουμε ενσωματωμένα λειτουργικά συστήματα όπως το eCos ή το uC/OS για δίκτυα αισθητήρων. Ωστόσο, τέτοια λειτουργικά συστήματα είναι συχνά σχεδιασμένα με ιδιότητες πραγματικού χρόνου. Τέτοια είναι:

- Το **TinyOS** είναι ίσως το πρώτο λειτουργικό σύστημα που σχεδιάστηκε ειδικά για ασύρματα δίκτυα αισθητήρων. Το TinyOS είναι βασισμένο πάνω σε event – driven programming μοντέλο αντί multithreading. Τα προγράμματα του TinyOS τα συνθέτουν event – handlers και tasks με run – to – completition semantics. Όταν ένα εξωτερικό γεγονός συμβεί, όπως εισερχόμενο πακέτο δεδομένων ή ένδειξη από αισθητήρα, το TinyOS δίνει σήμα στον κατάλληλο event – handler για να διαχειριστεί το event. Οι event – handlers μπορούν να αναβάλουν tasks που είναι προγραμματισμένα από τον TinyOS kernel για αργότερα.
- Το **LifeOS** είναι ένα νέο λειτουργικό σύστημα για ασύρματα δίκτυα αισθητήρων, το οποίο παρέχει abstraction σαν του UNIX και υποστήριξη για την γλώσσα προγραμματισμού C.
- Το **Contiki** είναι ένα λειτουργικό σύστημα το οποίο χρησιμοποιεί ένα απλούστερο προγραμματιστικό στυλ σε C ενώ παρέχει 6LoWPAN και Protothreads.

- **Online συνεργασία πλατφόρμων διαχείρησης δεδομένων αισθητήρα:**

Οι Online συνεργασία πλατφόρμες διαχείρισης δεδομένων αισθητήρα είναι on – line υπηρεσίες βάσης δεδομένων που επιτρέπουν στου ιδιοκτήτες αισθητήρων να καταγράφουν και να συνδέουν τις συσκευές για να τροφοδοτούν τα δεδομένα σε μία online βάση δεδομένων για αποθήκευση και επίσης επιτρέπουν σε προγραμματιστές να συνδέονται στην βάση δεδομένων και να φτιάχνουν τις δικές τους εφαρμογές βασισμένες σε αυτά τα δεδομένα.

Παραδείγματα είναι οι Xively και Wikisensing πλατφόρμες. Τέτοιες πλατφόρμες απλοποιούν την online συνεργασία μεταξύ των χρηστών πάνω σε ποικίλα σετ δεδομένων που κυμαίνονται από ενεργειακά και περιβαλλοντικά δεδομένα έως αυτά που έχουν συλλεχθεί από υπηρεσίες μεταφοράς. Άλλες υπηρεσίες περιλαμβάνουν: τη δυνατότητα σε προγραμματιστές να ενσωματώνουν γραφήματα χρόνου και widgets σε websites, την δυνατότητα να αναλύουν και να επεξεργάζονται δεδομένα ιστορικού αποσπασμένα από άλλα data feeds, να αποστέλλουν ειδοποιήσεις πραγματικού χρόνου από οποιοδήποτε datastream για να ελέγχουν scripts, συσκευές και περιβάλλοντα.

2.6.5 Εφαρμογές WSN

Παρακολούθηση Περιοχής

Η παρακολούθηση περιοχής είναι μια κοινή εφαρμογή των αισθητηριακών δικτύων. Στην παρακολούθηση περιοχής, το ασύρματο δίκτυο αισθητήρων έχει αναπτυχθεί σε μια περιοχή όπου κάποιο φαινόμενο πρέπει να παρακολουθηθεί. Ένα παράδειγμα από τον στρατό είναι η χρήση των αισθητήρων ώστε να ανιχνευθεί η εχθρική εισβολή. Ένα πολιτικό παράδειγμα είναι η γεωπερίφραξη του φυσικού αερίου ή στους αγωγούς πετρελαίου.

Παρακολούθηση Υγείας

Τα δίκτυα αισθητήρων για ιατρικές εφαρμογές μπορούν να είναι διάφορων τύπων: εμφυτευμένα, φορετά (wearable) και ενσωματωμένα στο περιβάλλον. Οι συσκευές – εμφυτεύματα είναι αυτές που μπαίνουν μέσα στο ανθρώπινο σώμα. Οι wearable συσκευές χρησιμοποιούνται στην επιφάνεια του ανθρώπινου σώματος ή σε κοντινή απόσταση από τον χρήστη. Τα ενσωματωμένα στο περιβάλλον συστήματα χρησιμοποιούν αισθητήρες που εμπεριέχονται στο περιβάλλον. Πιθανές εφαρμογές περιλαμβάνουν την μέτρηση θέσης σώματος, τοποθεσία ατόμων, εξ' ολοκλήρου παρακολούθηση ασθενών σε νοσοκομεία ή στο

σπίτι. Οι ενσωματωμένες συσκευές στο περιβάλλον ανιχνεύουν την φυσική κατάσταση ενός ατόμου για συνεχή διάγνωση υγείας, χρησιμοποιώντας σαν είσοδο δεδομένα από ένα δίκτυο με κάμερες βάθους, αισθητήριο πάτωμα ή άλλες παρόμοιες συσκευές. Τα body – area networks μπορούν να συλλέξουν πληροφορίες για την υγεία και την δαπάνη ενέργειας ενός ατόμου. Σε τέτοιες εφαρμογές τα προσωπικά δεδομένα και η πιστοποίηση των δεδομένων του χρήστη χρίζουν μεγάλης σημασίας. Ειδικά λόγω της ενσωμάτωσης των δικτύων αισθητήρων με το Internet of Things, η πιστοποίηση ενός χρήστη αποτελεί μεγαλύτερη πρόκληση.

Περιβαλλοντική / Γεωσκόπηση

Ο όρος Περιβαλλοντικά Δίκτυα Αισθητήρων έχει εξελιχθεί για να καλύψει πολλές εφαρμογές των ασύρματων δικτύων αισθητήρων για την έρευνα της γεωλογίας. Αυτό περιλαμβάνει την παρακολούθηση με αισθητήρες ηφαιστείων, ωκεανών, παγετώνων, δασών κτλ. Ορισμένοι από τους κύριους τομείς αναφέρονται παρακάτω.

- **Παρακολούθηση της ρύπανσης του αέρα:** Ασύρματα δίκτυα αισθητήρων έχουν αναπτυχθεί σε διάφορες πόλεις (Στοκχόλμη, Λονδίνο και Μπρισμπέιν) για την παρακολούθηση της συγκέντρωσης των επικίνδυνων αερίων για τους πολίτες. Αυτά μπορούν να επωφεληθούν από τις ασύρματες ζεύξεις ad-hoc και όχι από τις ενσύρματες εγκαταστάσεις που επίσης τα κάνουν πιο ευκίνητα για δοκιμαστικές μετρήσεις σε διάφορες περιοχές. Υπάρχουν διάφορες αρχιτεκτονικές που μπορούν να χρησιμοποιηθούν για τέτοιες εφαρμογές, καθώς και διάφορα είδη ανάλυσης δεδομένων και εξόρυξης δεδομένων που μπορούν να διεξαχθούν.
- **Ανίχνευση δασικών πυρκαγιών:** Ένα δίκτυο αισθητήριων κόμβων μπορεί να εγκατασταθεί σε ένα δάσος για να ανιχνεύει πότε έχει εκδηλωθεί πυρκαγιά. Οι κόμβοι μπορούν να είναι εξοπλισμένοι με αισθητήρες για τη μέτρηση της θερμοκρασίας, την υγρασία και τα αέρια που παράγονται από φωτιά στα δέντρα ή τη βλάστηση. Η έγκαιρη ανίχνευση είναι ζωτικής σημασίας για την επιτυχή δράση των πυροσβεστών, χάρη στα ασύρματα δίκτυα αισθητήρων, η πυροσβεστική θα είναι σε θέση να γνωρίζει πότε μια πυρκαγιά ξεκίνησε και πώς εξαπλώνεται.
- **Ανίχνευση κατολισθήσεων:** Ένα σύστημα ανίχνευσης κατολίσθησης κάνει χρήση ενός ασύρματου δικτύου αισθητήρων για να ανιχνεύσει τις μικρές κινήσεις του εδάφους και αλλαγές στις διάφορες παραμέτρους που μπορεί να συμβούν πριν ή κατά τη διάρκεια μιας κατολίσθησης. Μέσα από τα δεδομένα που συλλέγονται μπορεί να είναι δυνατόν να γνωρίζουμε την εμφάνιση των κατολισθήσεων πολύ πριν αυτό συμβεί στην πραγματικότητα.
- **Παρακολούθηση της ποιότητας των υδάτων:** Η παρακολούθηση της ποιότητας του νερού περιλαμβάνει την ανάλυση των ιδιοτήτων του νερού σε φράγματα, ποτάμια, λίμνες και ωκεανούς, καθώς και τα υπόγεια αποθέματα νερού. Η χρήση πολλών ασύρματων αισθητήρων που διανέμονται επιτρέπει τη δημιουργία μιας πιο ακριβούς εικόνας της κατάστασης των υδάτων, και επιτρέπει τη μόνιμη εγκατάσταση σταθμών παρακολούθησης σε περιοχές με δύσκολη πρόσβαση, χωρίς την ανάγκη του εγχειριδίου ανάκτησης δεδομένων.
- **Πρόληψη φυσικών καταστροφών:** Τα ασύρματα δίκτυα αισθητήρων μπορούν να ενεργήσουν αποτελεσματικά για να αποτραπούν οι συνέπειες των φυσικών καταστροφών, όπως οι πλημμύρες. Ασύρματοι κόμβοι έχουν αναπτυχθεί με επιτυχία σε ποτάμια όπου οι μεταβολές της στάθμης του νερού θα πρέπει να παρακολουθείτε σε πραγματικό χρόνο.

Βιομηχανική παρακολούθηση

Ασύρματα δίκτυα αισθητήρων έχουν αναπτυχθεί για την βασική συντήρηση των μηχανημάτων (Condition-Based Maintenance - CBM), δεδομένου ότι προσφέρουν σημαντική εξοικονόμηση κόστους και επιτρέπουν νέες λειτουργίες. Σε ενσύρματα συστήματα, η εγκατάσταση των αισθητήρων συχνά περιορίζεται από το κόστος της καλωδίωσης. Προηγουμένως απρόσιτες περιοχές, περιστρεφόμενα μηχανήματα, επικίνδυνες ή ζώνες περιορισμένης πρόσβασης και τα κινητά περιουσιακά στοιχεία μπορούν πλέον να επιτευχθούν με ασύρματους αισθητήρες.

- **Παρακολούθηση υγείας μηχανών:** Ασύρματα δίκτυα αισθητήρων έχουν αναπτυχθεί για την συντήρηση των μηχανημάτων με βάση την κατάστασή τους καθώς προσφέρουν σημαντική εξοικονόμηση κόστους. Ασύρματοι αισθητήρες μπορούν να τοποθετηθούν σε τοποθεσίες που είναι δύσκολο ή αδύνατον να προσεγγίσεις με ένα ενσύρματο σύστημα, όπως περιστρεφόμενα μηχανήματα.
- **Παρακολούθηση Data Center:** Λόγω της υψηλής πυκνότητας από server racks σε ένα data center, συχνά η καλωδίωση και οι διευθύνσεις IP αποτελούν πρόβλημα. Για να ξεπεραστεί αυτό το πρόβλημα όλο και περισσότερα racks εξοπλίζονται με ασύρματους αισθητήρες θερμοκρασίας για να παρακολουθούν την εισαγόμενη και εξαγόμενη θερμοκρασία στα racks. Καθώς η ASHRAE προτείνει έως και 6 αισθητήρες θερμοκρασίας ανά rack, η τεχνολογία πλέγματος ασύρματων αισθητήρων δίνει ένα πλεονέκτημα σε σχέση με τους παραδοσιακούς ενσύρματους αισθητήρες.
- **Data logging:** Τα ασύρματα δίκτυα αισθητήρων συχνά χρησιμοποιούνται για την συλλογή δεδομένων για την παρακολούθηση περιβαλλοντικών πληροφοριών. Αυτό μπορεί να αποτελεί από κάτι τόσο απλό, όπως η παρακολούθηση της θερμοκρασίας ενός ψυγείου, μέχρι το επίπεδο του νερού σε δεξαμενές υπερχείλισης σε πυρηνικά εργοστάσια παραγωγής ηλεκτρικής ενέργειας. Οι στατιστικές αυτές πληροφορίες, μπορούν αργότερα να χρησιμοποιηθούν για να μας δείξουν πως δουλεύαν τα συστήματα μέχρι τώρα. Το μεγάλο πλεονέκτημα των ασύρματων δικτύων αισθητήρων είναι πως σε σχέση με τους συμβατικούς loggers, αυτά δίνουν ζωντανή μετάδοση δεδομένων.
- **Παρακολούθηση νερού/αποβλήτων υδάτων:** Η παρακολούθηση της ποιότητας και του επιπέδου του νερού περιλαμβάνει πολλές δραστηριότητες, όπως τον έλεγχο της ποιότητας των υπόγειων ή επιφανειακών υδάτων και την εξασφάλιση υποδομών ύδρευσης της χώρας, προς όφελος ανθρώπων και ζώων. Η περιοχή της παρακολούθησης της ποιότητας του νερού.
- **Παρακολούθηση πολιτικών δομών:** Ασύρματα δίκτυα αισθητήρων μπορούν να χρησιμοποιηθούν για την παρακολούθηση της κατάστασης πολιτικών υποδομών και σχετικές γεο – φυσικές διεργασίες σχεδόν σε πραγματικό χρόνο και για μεγάλα χρονικά διαστήματα μέσω του data logging χρησιμοποιώντας κατάλληλα συνδεδεμένους αισθητήρες.
- **Παραγωγή κρασιού:** Ασύρματα δίκτυα αισθητήρων χρησιμοποιούνται για την παρακολούθηση της παραγωγής του κρασιού, τόσο και στο χωράφι όσο και στο κελάρι.

2.6.6 Περιορισμοί

1) Κατέχει πολύ μικρή αποθηκευτική χωρητικότητα – μερικές εκατοντάδες kilobytes.
2) Κατέχει μέτρια επεξεργαστική ισχύ – 8MHz.
3) Λειτουργεί σε μικρής εμβέλειας πεδίο επικοινωνίας – καταναλώνει αρκετή ενέργεια.
4) Απαιτεί ελάχιστη ενέργεια – περιορίζει τα πρωτόκολλα.
5) Έχει μπαταρίες με πεπερασμένη διάρκεια ζωής.
6) Οι παθητικές συσκευές παρέχουν λίγη ενέργεια.

2.7 Εφαρμογές IoT

Ο τομέας των εφαρμογών στο Internet of Things είναι εκτενής. Πολλαπλές κατηγοριοποιήσεις έχουν προταθεί, οι περισσότερες από τις οποίες συμφωνούν το διαχωρισμό μεταξύ εφαρμογών για το ευρύ κοινό, για επιχειρήσεις και για υποδομές.

Η δυνατότητα δικτύωσης ενσωματωμένων συσκευών με περιορισμένη επεξεργαστική ισχύ, μνήμη και πόρους ενέργειας σημαίνει ότι το IoT βρίσκει εφαρμογή σχεδόν σε κάθε πεδίο. Τέτοια συστήματα θα μπορούσαν να είναι υπεύθυνα για την συλλογή πληροφοριών σε τοποθετήσεις που κυμαίνονται από φυσικά οικοσυστήματα μέχρι κτήρια και εργοστάσια, βρίσκοντας έτσι εφαρμογές στους τομείς της περιβαλλοντικής ανίχνευσης και του πολεοδομικού σχεδιασμού.

Τα ευφυή συστήματα αγορών, για παράδειγμα, θα μπορούσαν να παρακολουθούν τις συνήθεις αγορές των χρηστών από την ανίχνευση των κινητών τους τηλεφώνων. Αυτοί οι χρήστες θα μπορούσαν να λάβουν ειδικές προσφορές για τα αγαπημένα τους προϊόντα ή ακόμα και την τοποθεσία των πραγμάτων που χρειάζονται, την οποία, για παράδειγμα, το έξυπνο ψυγείο θα έχει μεταβιβάσει αυτόματα στο κινητό του χρήστη. Επιπλέον παραδείγματα ανίχνευσης και ενεργοποίησης αντικατοπτρίζονται σε εφαρμογές που ασχολούνται με τη διαχείριση της θερμότητας, του νερού, της ηλεκτρικής ενέργειας και της διαχείρισης ενέργειας, όπως και βοηθητικά συστήματα πλοήγησης για τις μεταφορές. Άλλες εφαρμογές που το Internet of Things μπορεί να παρέχει είναι σαφώς και η εκτενής οικιακή ασφάλεια και οι οικιακοί αυτοματισμοί. Η έννοια ενός «Internet of living things» έχει προταθεί για την περιγραφή δικτύων που αποτελούνται από βιολογικούς αισθητήρες που θα μπορούσαν να χρησιμοποιήσουν αναλύσεις βασισμένες σε cloud για να επιτρέπουν στου χρήστες να μελετάνε το DNA ή άλλα μόρια.

2.7.1 Εφαρμογές Καταναλωτών

Ένα όλο αυξανόμενο τμήμα των συσκευών IoT δημιουργούνται για χρήση από τους καταναλωτές. Παραδείγματα εφαρμογών για τους καταναλωτές περιλαμβάνουν το συνδεδεμένο αυτοκίνητο, ψυχαγωγία, οικιακή αυτοματοποίηση (επίσης γνωστή ως έξυπνες οικιακές συσκευές), φορετή τεχνολογία (wearables), συνδεδεμένη υγεία και συσκευές όπως πλυντήρια / στεγνωτήρια, ρομποτικές ηλεκτρικές σκούπες, καθαριστές αέρα, φούρνοι ή ψυγεία / καταψύκτες που χρησιμοποιούν Wi-Fi για απομακρυσμένη παρακολούθηση.

Οικιακοί Αυτοματισμοί

Οικιακός αυτοματισμός είναι είναι η διαδικασία ελέγχου των οικιακών συσκευών αυτόματα χρησιμοποιώντας διάφορες τεχνικές ελέγχου συστήματος. Οι ηλεκτρικές και ηλεκτρονικές συσκευές στο σπίτι, όπως ο ανεμιστήρας, τα φώτα, τα εξωτερικά φώτα, ο συναγερμός πυρκαγιάς, ο χρονοδιακόπτης κουζίνας κ.λπ., μπορούν να ελέγχονται χρησιμοποιώντας διάφορες τεχνικές ελέγχου. Ο ασύρματος αυτοματισμός του σπιτιού που χρησιμοποιεί το IoT είναι μια καινοτόμος εφαρμογή του διαδικτύου που αναπτύχθηκε για τον έλεγχο των οικιακών συσκευών εξ αποστάσεως μέσω του cloud.

Τα βασικά εξαρτήματα και τα υλικά για οικιακό αυτοματισμό που χρησιμοποιούν το έργο IoT μπορούν να καταχωρηθούν ως μονάδα Wi-Fi, Opto-coupler, TRIAC, αντιστάσεις, πυκνωτές, δίοδοι, ρυθμιστές και οικιακές συσκευές.

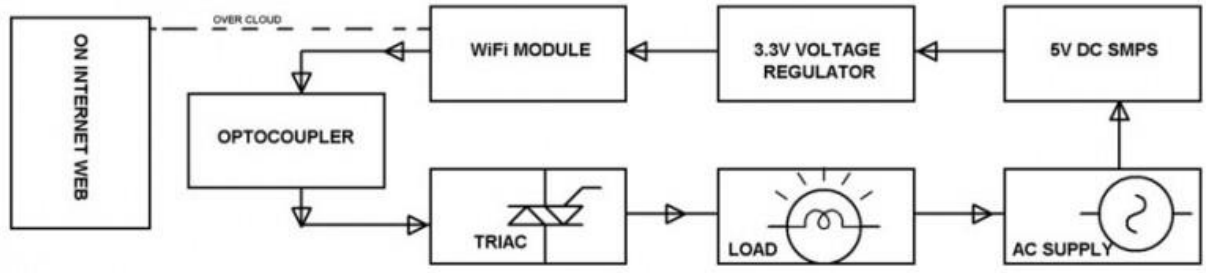

Figure 2-12 IoT Block Diagram

Το φορτίο (οικιακές συσκευές) μπορεί να ελεγχθεί και να παρακολουθηθεί χρησιμοποιώντας μια ιστοσελίδα με ρυθμιζόμενο από το χρήστη front – end. Ο χρήστης μπορεί να στείλει εντολές μέσω της παραχωρημένης διεύθυνσης IP και αυτές οι εντολές τροφοδοτούνται στη μονάδα Wi-Fi. Η μονάδα Wi-Fi έχει διαμορφωθεί για πρόσβαση στο διαδίκτυο χρησιμοποιώντας οποιοδήποτε κοντινό ασύρματο μόντεμ. Οι εντολές που λαμβάνονται από μια μονάδα Wi-Fi εκτελούνται από ένα πρόγραμμα μέσα σε μια μονάδα Wi-Fi. Η μονάδα Wi-Fi συνδέεται με το TRIAC & Optocoupler, μέσω των οποίων τα φορτία ενεργοποιούνται και απενεργοποιούνται βάσει εντολών. Η κατάσταση φορτίου (ON ή OFF) θα εμφανιστεί στην ιστοσελίδα.

2.7.2 Εφαρμογές Επιχειρήσεων

Ο όρος «Enterprise IoT» ή EIoT, χρησιμοποιείται για την αναφορά όλων των συσκευών που χρησιμοποιούνται σε επιχειρησιακές και εταιρικές τοποθετήσεις. Μέχρι το 2019, υπολογίζεται ότι το EIoT θα απαριθμεί σχεδόν το 40% των συσκευών ή αλλιώς 9,1 δισεκατομμύρια από αυτές.

Μέσα Ενημέρωσης

Η χρήση του Internet of Things από τα μέσα μαζικής ενημέρωσης είναι κυρίως γύρω από το μάρκετινγκ και την μελέτη των συνηθειών των καταναλωτών. Μέσω της στοχοθετημένης συμπεριφοράς αυτές οι συσκευές συλλέγουν πολλά σημεία πληροφόρησης για εκατομμύρια άτομα. Χρησιμοποιώντας τα προφίλ που φτιάχτηκαν κατά τη διάρκεια της διαδικασίας στοχοθέτησης, οι παραγωγοι των μέσων ενημέρωσης προβάλλουν διαφήμιση που ταυτοποιείται με τις γνωστές συνήθειες του καταναλωτή σε μία συγκεκριμένη ώρα και τοποθεσία για την μεγιστοποίηση της επίδρασης. Επιπλέον πληροφορίες συλλέγονται από την ανίχνευση της διάδρασης του καταναλωτή με το προϊόν. Αυτό επιτυγχάνεται μέσω του conversion tracking (ανίχνευση μετατροπών), drop off rate, click through rate, ρυθμό εγγραφής και ρυθμό διάδρασης. Το μέγεθος των δεδομένων συχνά παρουσιάζει προκλήσεις καθώς είναι αρκετά μεγάλο. Ωστόσο τα πλεονεκτήματα που αποκτούνται από την αποθήκευση όλων αυτών των δεδομένων υπερέχουν αυτών των προκλήσεων.

2.7.3 Διαχείρηση Υποδομών

Η παρακολούθηση και ο έλεγχος πολιτικών και αγροτικών υποδομών, όπως γέφυρες, σιδηροδρομικές γραμμές, παράκτιων και μη περιοχών με ανεμογεννήτριες είναι εφαρμογή κλειδί για το IoT. Η IoT υποδομή μπορεί να χρησιμοποιηθεί για την παρακολούθηση οποιονδήποτε γεγονότων ή αλλαγών στις καταστάσεις δομικών έργων που μπορούν να εκθέσουν την ασφάλεια και να αυξήσουν το ρίσκο. Μπορεί επίσης να χρησιμοποιηθεί για προγραμματισμό συντήρησης και δραστηριότητες διατήρησης αποτελεσματικά, συντονίζοντας τα καθήκοντα μεταξύ διάφορων παρόχων υπηρεσιών και χρηστών αυτών των εγκαταστάσεων. Οι IoT συσκευές μπορούν επίσης να χρησιμοποιηθούν για τον έλεγχο κρίσιμων υποδομών

όπως γέφυρες για την παροχή πρόσβασης σε πλοία. Η χρήση των IoT συσκευών για παρακολούθηση και την λειτουργεία των υποδομών είναι πιθανόν να βελτιώσει την διαχείριση συμβάντων και τον συντονισμό απόκρισης σε έκτακτες ανάγκες, την ποιότητα υπηρεσιών και μειώνει το κόστος των λειτουργειών σε όλες τους τομείς που σχετίζονται με υποδομές. Ακόμα και ο τομέας την διαχείρισης λυμάτων θα μπορούσε να επωφεληθεί από μία IoT εγκατάσταση.

Γεωργία

Το IoT συνεισφέρει σημαντικά στις καινοτόμες αγροτικές μεθόδους. Οι γεωργικές προκλήσεις που προκαλούνται από την πληθυσμιακή ανάπτυξη και την αλλαγή του κλίματος έχουν καταστήσει αυτή τη βιομηχανία μια από τις πρώτες που χρησιμοποιούν το IoT. Η ενσωμάτωση ασύρματων αισθητήρων σε κινητές εφαρμογές και πλατφόρμες cloud βοηθά στην συλλογή ζωτικών πληροφοριών σχετικά με τις περιβαλλοντικές συνθήκες – η θερμοκρασία, η βροχόπτωση, η υγρασία, η ταχύτητα του ανέμου, η μόλυνση από τα παράσιτα, η περιεκτικότητα σε χούμο στο έδαφος ή τα θρεπτικά συστατικά μεταξύ άλλων – που συνδέονται με μία αγροτική περιοχή, που μπορούν να χρησιμοποιηθούν για τη βελτίωση και την αυτοματοποίηση των γεωργικών τεχνικών, τη λήψη τεκμηριωμένων αποφάσεων για τη βελτίωση της ποιότητας και της ποσότητας και την ελαχιστοποίηση των κινδύνων και των αποβλήτων. Επίσης μειώνονται οι δυσκολίες διαχείρισης καλλιεργειών σε πολλαπλές τοποθεσίες.

Διαχείριση Ενέργειας

Εκτός από την ενεργειακή διαχείριση που βασίζεται στο σπίτι, το IoT έχει ιδιαίτερη σημασία για το Smart Grid, δεδομένου ότι παρέχει συστήματα για τη συγκέντρωση και την δράση στην ενέργεια, και πληροφορίες σχετικές με την ενέργεια , με στόχο τη βελτίωση της αποδοτικότητας, της αξιοπιστίας, της οικονομίας και της βιωσιμότητας της παραγωγής και διανομής της ηλεκτρικής ενέργειας. Χρησιμοποιώντας συσκευές προηγμένης μετρητικής υποδομής (AMI) συνδεδεμένες με τη ραχοκοκαλιά του Internet, τα ηλεκτρικά βοηθητικά προγράμματα δεν θα μπορούν μόνο να συλλέγουν δεδομένα από συνδέσεις τελικών χρηστών, αλλά και να διαχειρίζονται άλλες συσκευές αυτοματοποίησης διανομής, όπως μετασχηματιστές και reclosers.

Εγκαταστάσεις μητροπολιτικής κλίμακας

Υπάρχουν αρκετές σχεδιασμένες ή τρέχουσες εκτενής αναπτύξεις του IoT για την καλύτερη διαχείριση των πόλεων και των συστημάτων τους. Για παράδειγμα, η Songdo στη Νότια Κορέα, η πρώτη από τις πλήρως εξοπλισμένες έξυπνες πόλεις της χώρας, βρίσκεται σχεδόν στην ολοκλήρωση την έννοιας smart city. Σχεδόν τα πάντα σε αυτήν την πόλη σχεδιάζονται για να συνδεθούν και να μετατραπούν σε μια συνεχή ροή δεδομένων που θα παρακολουθούνται και θα αναλύονται από μια σειρά ηλεκτρονικών υπολογιστών με μικρή ή και καθόλου ανθρώπινη παρέμβαση.

Μια άλλη εφαρμογή είναι ένα υπό εξέλιξη έργο στο Santander της Ισπανίας. Για την ανάπτυξη αυτή, έχουν υιοθετηθεί δύο προσεγγίσεις. Αυτή η πόλη με 180.000 κατοίκους έχει ήδη δει 18.000 λήψεις της εφαρμογής smartphone της πόλης της. Η εφαρμογή συνδέεται με 10.000 αισθητήρες που επιτρέπουν υπηρεσίες όπως την αναζήτηση χώρου στάθμευσης, την παρακολούθηση του περιβάλλοντος, ατζέντα ψηφιακής πόλης και πολλά άλλα. Στο πλαίσιο αυτής της ανάπτυξης χρησιμοποιούνται πληροφορίες σχετικά με την πόλη, έτσι ώστε να επωφελούνται οι έμποροι μέσω ενός μηχανισμού βασισμένου στη συμπεριφορά της πόλης που στοχεύει στη μεγιστοποίηση του αντίκτυπου του κάθε notification.

Άλλα παραδείγματα εκτεταμένων αναπτύξεων περιλαμβάνουν την Guangzhou Knowledge City, όπου το σύστημα δουλεύει για τη βελτίωση της ποιότητας του αέρα και των

υδάτων, τη μείωση της ηχορύπανσης και την αύξηση της αποδοτικότητας των μεταφορών στο Σαν Χοσέ της Καλιφόρνιας και την έξυπνη διαχείριση κυκλοφορίας στη δυτική Σιγκαπούρη. Η γαλλική εταιρεία Sigfox άρχισε να κατασκευάζει ένα ασύρματο δίκτυο δεδομένων στενής ζώνης στην Περιοχή του Κόλπου του Σαν Φρανσίσκο το 2014, η πρώτη επιχείρηση που πέτυχε μια τέτοια ανάπτυξη στις ΗΠΑ. Στη συνέχεια ανακοίνωσε ότι θα δημιουργήσει συνολικά 4000 σταθμούς βάσης για να καλύψει συνολικά 30 πόλεις στις Η.Π.Α. μέχρι το τέλος του 2016, καθιστώντας την ως τον μεγαλύτερο πάροχο κάλυψης δικτύου IoT στη χώρα μέχρι στιγμής.

Ένα άλλο παράδειγμα μεγάλης ανάπτυξης είναι εκείνο που ολοκληρώθηκε από τη New York Waterways στη Νέα Υόρκη για να συνδέσει όλα τα πλοία της πόλης και να μπορεί να τα παρακολουθεί ζωντανά όλο το 24ωρο. Το δίκτυο σχεδιάστηκε και κατασκευάστηκε από την Fluidmesh Networks, εταιρεία που εδρεύει στο Σικάγο και αναπτύσσει ασύρματα δίκτυα για κρίσιμες εφαρμογές. Το δίκτυο NYWW παρέχει επί του παρόντος κάλυψη στον ποταμό Hudson, τον East River και τον κόλπο της Άνω Νέας Υόρκης. Με το υπάρχον ασύρματο δίκτυο, η NY Waterway είναι σε θέση να αναλάβει τον έλεγχο του στόλου και των επιβατών της με τρόπο που δεν ήταν εφικτός. Οι νέες εφαρμογές μπορούν να περιλαμβάνουν ασφάλεια, ενέργεια και διαχείριση στόλου, ψηφιακή σήμανση, δημόσιο Wi-Fi, έκδοση ηλεκτρονικών εισιτηρίων και άλλα.

2.7.4 Άλλα Πεδία Εφαρμογής

Ιατρική και Υγειονομική Περίθαλψη

Οι συσκευές IoT μπορούν να χρησιμοποιηθούν για τη δυνατότητα απομακρυσμένης παρακολούθησης της υγείας και συστημάτων ειδοποίησης έκτακτης ανάγκης. Αυτές οι συσκευές παρακολούθησης της υγείας μπορούν να κυμαίνονται από τους μετρητές της αρτηριακής πίεσης και του καρδιακού ρυθμού έως τις προηγμένες συσκευές που είναι σε θέση να παρακολουθούν εξειδικευμένα εμφυτεύματα, όπως βηματοδότες, ηλεκτρονικά βραχιόλια Fitbit ή προηγμένα ακουστικά βοηθήματα. Ορισμένα νοσοκομεία έχουν αρχίσει να εφαρμόζουν «έξυπνα κρεβάτια» που μπορούν να ανιχνεύσουν όταν είναι κατειλημμένα και όταν ένας ασθενής προσπαθεί να σηκωθεί. Μπορούν επίσης να προσαρμοστούν για να εξασφαλίσουν την κατάλληλη πίεση και υποστήριξη για τον ασθενή χωρίς τη χειρωνακτική αλληλεπίδραση των νοσοκόμων.

Μεταφορά

Το IoT μπορεί να βοηθήσει στην ενοποίηση των επικοινωνιών, του ελέγχου και της επεξεργασίας πληροφοριών σε διάφορα συστήματα μεταφοράς. Η εφαρμογή του IoT επεκτείνεται σε όλες τις πτυχές των συστημάτων μεταφοράς (δηλαδή του οχήματος, της υποδομής και του οδηγού ή του χρήστη). Η δυναμική αλληλεπίδραση μεταξύ αυτών των στοιχείων ενός συστήματος μεταφορών επιτρέπει την ενδοεπικοινωνία με τα οχήματα, τον έξυπνο έλεγχο της κυκλοφορίας, τον έξυπνο χώρο στάθμευσης, τα ηλεκτρονικά συστήματα είσπραξης διοδίων, την υλικοτεχνική διαχείριση και τη διαχείριση του στόλου, τον έλεγχο των οχημάτων και την ασφάλεια και την οδική βοήθεια. Για παράδειγμα, η πλατφόρμα IoT μπορεί να παρακολουθεί συνεχώς τη θέση και τις συνθήκες φορτηγών κ.α μέσω ασύρματων αισθητήρων και να αποστέλλει συγκεκριμένες ειδοποιήσεις όταν προκύπτουν εξαιρέσεις διαχείρισης (καθυστερήσεις, ζημιές, κλοπές κλπ.).

3 OpenHAB 2

Figure 3-1 openHAB 2 logo

3.1 Τι είναι

Το openHAB αποτελεί ακρωνύμιο του **open** **H**ome **A**utomation **B**us και είναι μία ανοιχτού κώδικα τεχνολογική πλατφόρμα για την υλοποίηση οικιακών αυτοματισμών. Το λογισμικό αυτό ενσωματώνει διάφορα συστήματα, συσκευές και τεχνολογίες οικιακών αυτοματισμών σε μία ενιαία λύση. Προσφέρει ομοιόμορφες διεπαφές χρήστη, και μία κοινή προσέγγιση σε κανόνες αυτοματισμού σε ολόκληρο το σύστημα, ανεξάρτητα από τον αριθμό των κατασκευαστών και των υποσυστημάτων που το απαρτίζουν.

3.2 Δομή openHAB

Το OpenHAB είναι ανεπτυγμένο σε Java και κυρίως βασισμένο στο Eclipse SmartHome framework. Χρησιμοποιεί Apache Karaf μαζί με το Eclipse Equinox για να δημιουργήσει το περιβάλλον εκτέλεσης Open Services Gateway initiative (OSGi). Το Jetty χρησιμοποιείται για HTTP server.

Το OpenHAB, καθώς αποτελεί ένα «σπονδυλωτό» (modular) λογισμικό, μπορεί να επεκταθεί με την χρήση “Add-ons”. Τα Add-ons δίνουν στο openHAB ένα ευρύ φάσμα δυνατοτήτων, από διεπαφές χρήστη, μέχρι την ικανότητα να διαδρά με ένα μεγάλο και αυξανόμενο αριθμό φυσικών αντικειμένων.

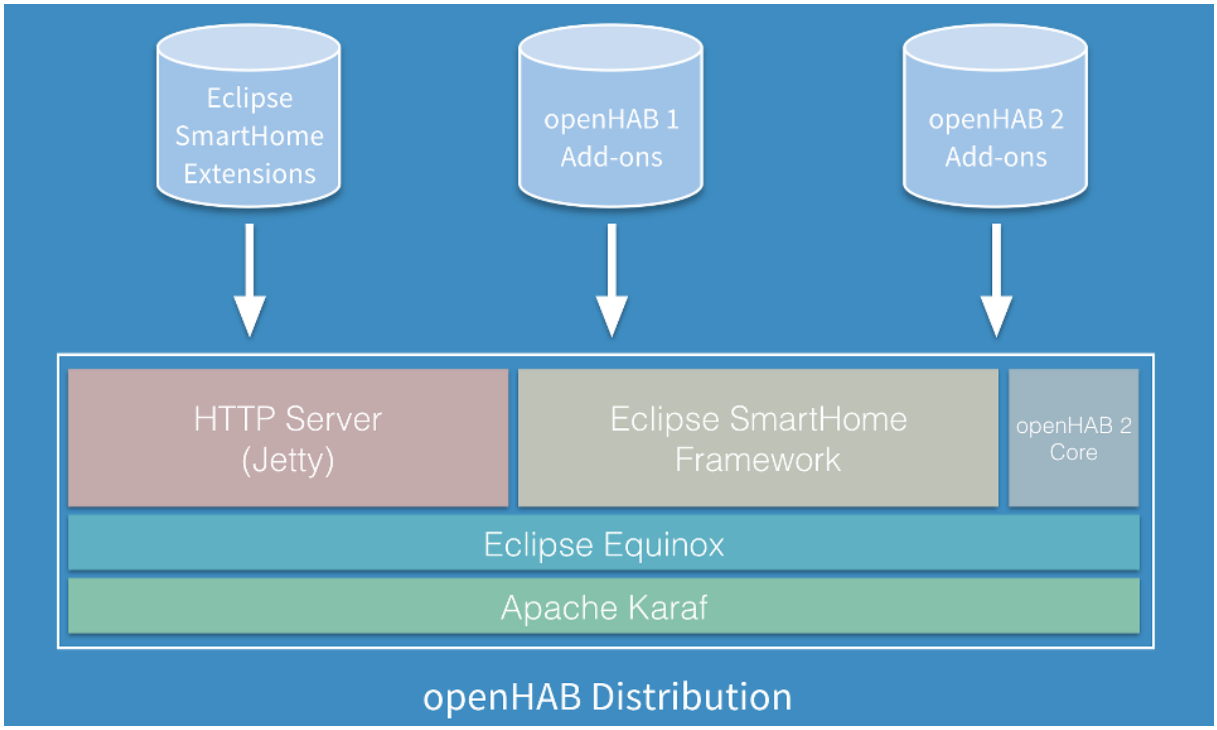

Figure 3-2 openHAB structure

3.3 Έννοιες

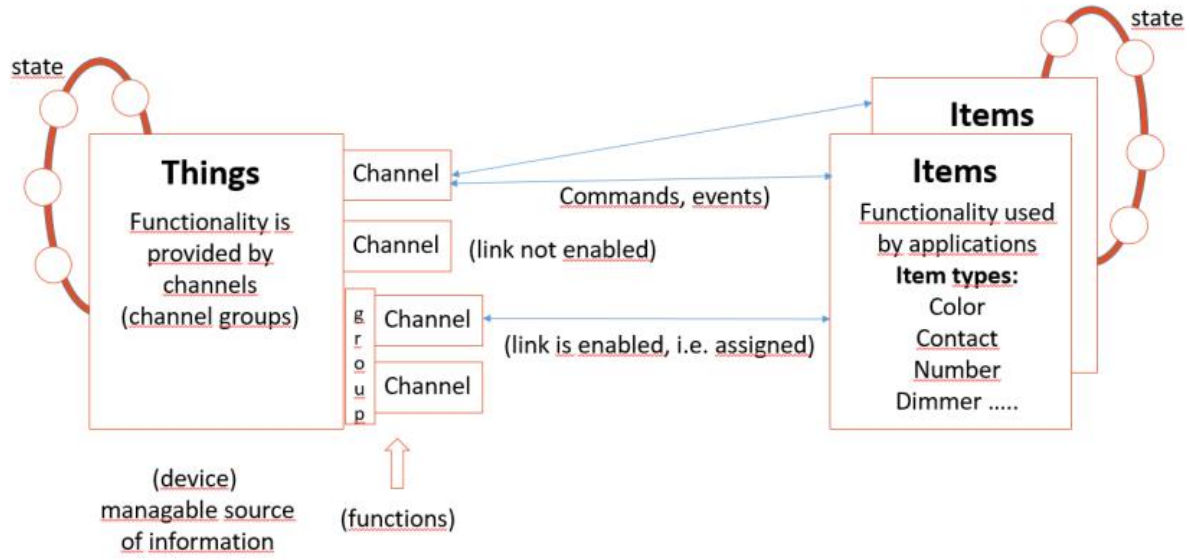

Figure 3-3 openHAB concepts

3.3.1 Σφαιρική εικόνα

Το Eclipse SmartHome διαφοροποιείται αυστηρά μεταξύ της φυσικής και της λειτουργικής άποψης του συστήματος. Ενώ η φυσική σκοπιά απαιτείται για την εγκατάσταση, διαμόρφωση, εντοπισμό προβλημάτων κ.λπ. του συστήματος, η λειτουργική καλύπτει τις πληροφορίες που έχουν σημασία στις εφαρμογές, όπως οι διεπαφές χρήστη και η λογική αυτοματισμού.

3.3.2 Things, Channels, Items and Links

Με τον όρο Things αναφερόμαστε στις οντότητες οι οποίες μπορούν να προστεθούν φυσικά σε ένα σύστημα και δυνητικά παρέχουν πολλές λειτουργίες ταυτόχρονα. Είναι σημαντικό να σημειωθεί πως τα Things δεν είναι ανάγκη να είναι συσκευές, αλλά μπορούν επίσης να αναπαριστούν μία διαδικτυακή υπηρεσία ή οποιαδήποτε άλλη διαχειρίσιμη πηγή πληροφορίας και λειτουργικότητας. Από την προοπτική του χρήστη, είναι σχετικά για την ρύθμιση και την διαδικασία διαμόρφωσης του συστήματος αλλά όχι για την λειτουργία του.

Τα Things παρέχουν την λειτουργικότητά τους μέσω μίας σειράς καναλιών (Channels). Τα κανάλια είναι παθητικά και μπορούν να θεωρηθούν ως μία δήλωση ενός Thing για το τι μπορεί να προσφέρει. Για παράδειγμα, ένας λαμπτήρας μπορεί να έχει ένα dimer channel και ένα color channel, όπου και τα δύο παρέχουν την λειτουργικότητα του λαμπτήρα (Thing) στο σύστημα.

Με τον όρο Items αναφερόμαστε στις οντότητες που αναπαριστούν μία λειτουργικότητα η οποία χρησιμοποιείται από εφαρμογές, όπως οι διεπαφές χρήστη ή την λογική αυτοματισμού. Τα Items βρίσκονται σε μία κατάσταση, και μπορούν να λάβουν εντολές προκειμένου αυτή να αλλάξει. Το openHAB παρέχει διάφορους τύπους Items όπως, Switch, Contact, Number κ.α.

Αυτό που ενώνει τα Things και τα Items είναι οι σύνδεσμοι (Links). Οι σύνδεσμοι είναι ενώσεις μεταξύ ακριβώς ενός Thing Channel και ενός Item. Αν ένα κανάλι είναι συνδεδεμένο με ένα Item, τότε το Item αυτό είναι ενεργοποιημένο, που σημαίνει ότι η λειτουργικότητα που το Item αναπαριστά, είναι διαχειρίσιμη μέσω του δοθέντος καναλιού. Τα κανάλια μπορούν να συνδεθούν με πολλαπλά Items και τα Items με πολλαπλά κανάλια.

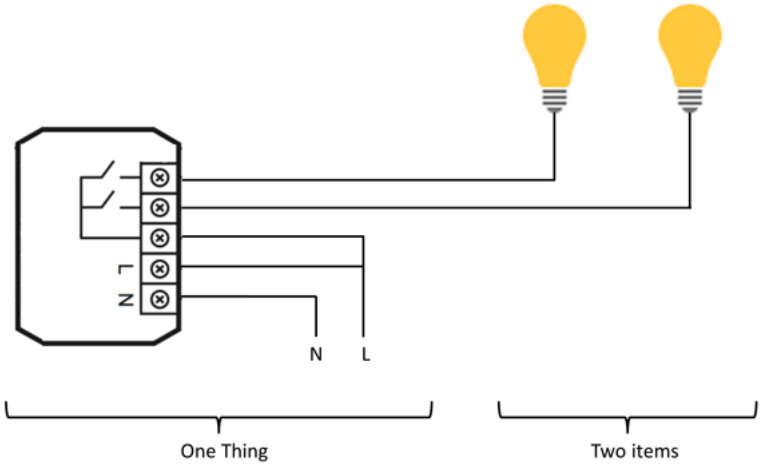

Figure 3-4 Παράδειγμα Things, Channels, Items and Links

Η παραπάνω εικόνα αποτελεί ένα παράδειγμα των εννοιών αυτών με τη χρήση ενός ενεργοποιητή και δύο λαμπτήρων. Ο ενεργοποιητής αποτελεί το Thing. Αυτός μπορεί να εγκατασταθεί σε ένα ηλεκτρικό πίνακα, έχε φυσική διεύθυνση και χρειάζεται να ρυθμιστεί και να διαμορφωθεί για να έχει τη δυνατότητα να χρησιμοποιηθεί. Ο χρήστης αντ' αυτού, ενδιαφέρεται για τους λαμπτήρες, οι οποίοι είναι τοποθετημένοι σε διαφορετικές τοποθεσίες μέσα στο σπίτι του. Αυτοί οι λαμπτήρες προσφέρουν την επιθυμητή λειτουργικότητα, άρα αποτελούν τα Items, και είναι συνδεδεμένοι με τα κανάλια του ενεργοποιητή. Ο σύνδεσμος (Link) μπορεί να θεωρηθεί σαν ένα πραγματικό καλώδιο σε αυτό το παράδειγμα.

3.4 Διαμόρφωση

3.4.1 Σφαιρική εικόνα

Το openHAB, λοιπόν, είναι ένα σύστημα που εγκαθίσταται και εκτελείται από τον χρήστη και τρέχει ανεξάρτητα από κάθε online υπηρεσίες και ιδιόκτητες τεχνολογίες με αποτέλεσμα ο τελικός χρήστης να έχει τον πλήρη έλεγχο πάνω σε κάθε όψη του οικιακού αυτοματισμού. Καθώς το openHAB είναι το κέντρο ενός οικιακού αυτοματισμού, όλες οι ιδιότητες και ικανότητες των συσκευών είναι διαθέσιμες στο user frontend και άλλα μέρη και συνδεδεμένα συστήματα, μέσω αυτού. Όμως κάθε συσκευή συνδεδεμένη στο openHAB είναι λειτουργικά και λογικά διαφορετική. Για την αναπαράσταση, λοιπόν, όλων αυτών των συσκευών, το openHAB ορίζει τα ακόλουθα βασικά μέρη:

- **Bindings** – Διάφορα Add-ons για την επικοινωνία με τις συσκευές.
- **Things** – Αναπαράσταση των συσκευών στο openHAB.
- **Items** – Ιδιότητες και ικανότητες των συσκευών.
- **Groups** – Συλλογές ή κατηγορίες που περιέχουν Items.
- **Sitemaps** – Διεπαφή που καθορίζεται από τον χρήστη για την διάταξη των Groups, Items κ.α
- **Transformations** – Βοηθητικές συναρτήσεις που μετατρέπουν τα δεδομένα.
- **Persistence** – Υπηρεσίες για την αποθήκευση δεδομένων με το πέρασμα του χρόνου.
- **Rules** – Λογική αυτοματισμού, κανόνες που ενεργοποιούνται από events, δηλαδή, την αλλαγή κατάστασης κάποιου Item.
- **JSR223 Scripting** – Ορισμός κανόνων και runtime objects με την χρήση Javascript, Jython ή Groovy.

Για την διαμόρφωση υπάρχει αρκετά μεγάλη ευελιξία καθώς το openHAB παρέχει αρκετές διεπαφές για την τροποποίηση των ρυθμίσεων, την διαχείριση των Things και των Items, την ανάπτυξη κανόνων και την πρόσβαση στο Sitemap.

Η διαμόρφωση μπορεί να γίνει είτε γραφικά είτε με χρήση text editor. Τα Things και τα Items μπορούν είτε να οριστούν και να διαχειριστούν μέσω των configuration files ή από το Paper UI σε μία βάση δεδομένων από τη μεριά του συστήματος. Το openHAB μας δίνει τη δυνατότητα χρήσης και των δύο μεθόδων ταυτόχρονα, αν το επιθυμούμε.

3.4.2 Configuration Files

Configuration files είναι τα αρχεία στα οποία ορίζουμε τις συσκευές μας σαν things ή items και τους κανόνες που τα διέπουν, ορίζουμε την εμφάνιση τους στο User Interface και τις διάφορες συναρτήσεις μετατροπής των δεδομένων. Ακόμα, ορίζουμε και τις υπηρεσίες που χρησιμοποιούμε (π.χ MQTT) και τα χαρακτηριστικά τους.

- **Ορισμός Things/Things Files**

  Όπως είπαμε τα Things αναπαριστούν το φυσικό επίπεδο σε ένα openHAB σύστημα. Από την άποψη διαμόρφωσης, τα Things λένε στο openHAB ποιες φυσικές οντότητες (συσκευές, web services, information sources, κ.λπ.) είναι για να διαχειριστούν από το σύστημα.

  Τα Things είναι συνδεδεμένα στο openHAB μέσω των bindings. Για να προσθέσει κάποιος ένα Thing στο σύστημα, πρέπει πρώτα να βρει και να εγκαταστήσει το κατάλληλο binding το οποίο θα παρέχει τον τύπο του Thing που θες να προσθέσεις. Για παράδειγμα, πριν προσθέσουμε σαν Thing μία συσκευή Z-Wave, το Z-Wave binding πρέπει να είναι εγκατεστημένο.

  Άρα για να ορίσουμε ένα Thing πρέπει να ακολουθήσουμε την εξής διαδικασία:

  - Εύρεση κατάλληλου binding
  - Εγκατάσταση binding
  - Ορισμός και διαμόρφωση Thing
  - Εύρεση των καναλιών του Thing
  - Προσθήκη Items και σύνδεση με το/τα κανάλι/α (επεξήγηση στον ορισμό Items)
  - Σε αυτό το σημείο τα Items μπορούν να χρησιμοποιηθούν για τον έλεγχο του Thing ή για την παροχή της πληροφορίας του στο sitemap ή τους κανόνες

  Για τον ορισμό ενός Thing υπάρχουν δύο τρόποι. Είτε με την αυτόματη «ανακάλυψη» του στο Paper UI, όπου το Thing εντοπίζεται αυτόματα από το openHAB και οι ιδιότητές του μπορούν να διαμορφωθούν από τον χρήστη μέσω της διεπαφής είτε με τον manual τρόπο όπου ορίζουμε το Thing στο αντίστοιχο configuration text file.

  Για να ορίσουμε ένα Thing χρησιμοποιώντας ένα configuration text file, πρέπει πρώτα να δημιουργήσουμε το αρχείο στον φάκελο things που βρίσκεται στο openHAB configuration path /etc/openhab2/things.Το αρχείο θα πρέπει να έχει την κατάληξη .thing. Η σύνταξη για τον ορισμό ενός Thing στο things configuration text file θα πρέπει να είναι η ως εξής:

```
Thing <binding_id>:<thing_id> “Label” @ “Location” [ <parameters> ]
```

- **Ορισμός Items/Items Files**

Τα Items αναπαριστούν όλες τις ιδιότητες και ικανότητες του οικιακού αυτοματισμού ενός χρήστη. Ενώ μία συσκευή ή υπηρεσία μπορεί να είναι αρκετά συγκεκριμένη, τα Items είναι ενοποιημένες υποκαταστάσεις του κόσμου του openHAB. Τα Items μπορούν να είναι τύπου String, Number, Switch ή κάποιου από τους άλλους βασικούς τύπους που παρέχει η πλατφόρμα. Ένας προγραμματιστής μπορεί να συγκρίνει τύπους Items μεταξύ τους με την χρήση βασικών μεταβλητών τύπων δεδομένων μίας γλώσσας προγραμματισμού.

Ένα αξιόλογο χαρακτηριστικό που μας παρέχει το openHAB είναι ότι τα Items έχουν τη δυνατότητα να συνδεθούν με τον «έξω κόσμο» μέσω των Bindings, καθώς ένα Item δεν αποθηκεύει απλά πληροφορία που καθορίζεται από κάποιο software, αλλά η πληροφορία αυτή μπορεί να οριστεί από διάφορες ενέργειες που λαμβάνουν χώρα μέσα στο σύστημα.

Τα Items, λοιπόν, είναι βασικοί τύποι δεδομένων και έχουν μία κατάσταση η οποία μπορεί να διαβαστεί αλλά και να αλλάξει. Μπορούν επίσης να είναι συνδεδεμένα με ένα Binding channel για να μπορούν να δρούν με το υπόλοιπο σύστημα. Για παράδειγμα ένα Item δεσμευμένο σε ένα αισθητήρα λαμβάνει ενημερώσεις από αυτόν και ένα Item συνδεδεμένο στο dimmer channel ενός λαμπτήρα, μπορεί να ρυθμίσει την φωτεινότητα της λάμπας.

Όπως και στα Things, υπάρχουν δύο τρόποι για να ορίσουμε ένα Item. Είτε χρησιμοποιώντας την διεπαφή του Paper UI, είτε με την χρήση configuration text files. Για τον ορισμό ενός Item με configuration text file, δημιουργούμε πρώτα το αρχείο στον αντίστοιχο φάκελο ο οποίος είναι /etc/openhab2/items και του δίνουμε την κατάληξη .items. Μπορούμε να έχουμε πάνω από ένα αρχείο τύπου .items αρκεί κάθε item που ορίζουμε σε αυτά να είναι μοναδικά. Η σύνταξη ενός Item μέσα στο configuration text file θα πρέπει είναι η ως εξής:

```
itemtype itemname "labeltext [stateformat]" <iconname> (group1)
                  ["tag1"] {bindingconfig}
```

Τα πεδία itemtype και itemname είναι υποχρεωτικά, μπορούν να υπάρξουν παραπάνω από ένα tag και group.

- **Sitemaps Files**

Στο openHAB μία συλλογή Things και Items αναπαριστά φυσικά ή λογικά αντικείμενα στο σύστημα. Τα sitemaps χρησιμοποιούνται για να επιλέξουν και να ετοιμάσουν αυτά τα στοιχεία προκειμένου να συνθέσουν μία φιλική προς τον χρήστη παρουσίαση αυτής της εγκατάστασης για διάφορες διεπαφές, συμπεριλαμβανομένου του Basic UI, της Android εφαρμογής κ.α.

Τα sitemaps είναι configuration text files με την κατάληξη .sitemap, και πρέπει να τα δημιουργήσουμε στον φάκελο /etc/openhab2/sitemaps. Η σύνταξη ενός sitemap αρχείου θα μπορούσε είναι η εξής:

```
Sitemap demo label=”My home automation”{
     Frame label=”Date”{
          Text item=”Date”
     }
     Frame label=”Demo”{
          Switch item=Lights icon=”light”
          Text item = LR_Temp label = “Room [%.1 ˚C]”
     }
}
```

- **Transformations Files**

Οι μετατροπές χρησιμοποιούνται για την μετάφραση δεδομένων από μία εξειδικευμένη ορολογία ή συντόμευση σε μία φιλική προς τον χρήστη μορφή. Για παράδειγμα μετατρέπουν εισερχόμενες τιμές ενός αισθητήρα για να μπορεί να τις καταλάβει ο χρήστης ή ακόμα και να μετατρέπει τιμές ενός Item και να τις κάνει κατανοητές για κάποιο άλλο Item.

Τα αρχεία για τις μετατροπές map (επεξήγηση στα Addons/transformation services) τα αποθηκεύουμε στον φάκελο /etc/openhab2/transform και τους δίνουμε την κατάληξη .map. Σημαντικό είναι να έχουμε εγκατεστημένα transformation services (πχ Map). Η σύνταξη και η λογική των αρχείων καθορίζεται από αυτήν. Παράδειγμα σύνταξης transformation map file:

```
0=No Connection
1=Bad Connection
2=Medium Connection
3=Good Connection
4=Strong Connection
```

- **Rules Files**

Οι κανόνες χρησιμοποιούνται για διαδικασίες αυτοματισμού· κάθε κανόνας μπορεί να ενεργοποιηθεί, ο οποίος με τη σειρά του επικαλείται ένα script που εκτελεί κάθε είδους δουλειάς, πχ αλλαγή κατάστασης κάποιου Item, μαθηματικές πράξεις, έναρξη χρονομετρητή κ.α. Το openHAB συμπεριλαμβάνει μία αρκετά ολοκληρωμένη, ελαφριά αλλά και δυνατή μηχανή κανόνων.

Ο ορισμός των κανόνων γίνεται σε ένα αρχείο που δημιουργούμε στην τοποθεσία /etc/openhab2/rules και με κατάληξη .rules. Ένα .rules αρχείο μπορεί να περιέχει πολλαπλούς κανόνες. Όλοι οι κανόνες ενός αρχείου μπορούν να έχουν πρόσβαση στις μεταβλητές μεταξύ τους και να τις ανταλλάσσουν. Το Eclipse SmartHome Designer προσφέρει υποστήριξη για την ανάπτυξη κανόνων. Περιέχει συντακτικό έλεγχο και χρωματισμό, επικύρωση με δείκτες σφάλματος (error markers), βοήθεια περιεχομένου (content assist) κ.α.

Η σύνταξη των κανόνων είναι βασισμένη στην Xbase και κατ’ επέκταση μοιράζεται πολλές λεπτομέρειες με την Xtend, η οποία είναι επίσης χτισμένη πάνω στην Xbase. Τέλος, ένα αρχείο κανόνων είναι ένα αρχείο κειμένου με την εξής μορφη:

- Imports
- Variable Declarations
- Rules

Στο τμήμα των Imports εμπεριέχονται, ακριβώς όπως στη Java, τα import statements τα οποία επιθυμούμε. Όπως στην Java, οι εισαγόμενοι τύποι μας είναι διαθέσιμοι χωρίς να χρειάζεται να γράψουμε το πλήρες όνομα τους. Μερικά imports έχουν ήδη γίνει οπότε δεν χρειάζεται να τα κάνουμε εμείς.

Το τμήμα της δήλωσης μεταβλητών (Variable Declaration) μπορεί να χρησιμοποιηθεί για να δηλώσουμε μεταβλητές που θέλουμε να είναι προσβάσιμες από όλους τους κανόνες του αρχείου. Μπορούμε να δηλώσουμε μεταβλητές με ή χωρίς αρχικές τιμές και τροποποιήσιμες ή μόνο για ανάγνωση.

Το τμήμα των κανόνων περιέχει μία λίστα από κανόνες. Κάθε κανόνας μπορεί να έχει πάνω από μία συνθήκες ενεργοποίησης. Κάτω από αυτές τις συνθήκες είναι το script block, όπου υπάρχει ο κώδικας που θα εκτελεστεί αν ενεργοποιηθούν οι συνθήκες. Υπάρχουν διάφορες κατηγορίες συνθηκών για την ενεργοποίηση των κανόνων. Αυτές είναι οι Item(-Event) – based, δηλαδή αντιδράσεις σε συμβάντα στο openHAB event bus, πχ εντολές και ενημέρωση καταστάσεων των Items, οι time – based, που αντιδρούν συγκεκριμένες ώρες, πχ μεσάνυχτα, κάθε ώρα κ.λπ., οι system – based, ενεργοποιούνται σε συγκεκριμένες καταστάσεις του συστήματος και οι thing – based, που αντιδρούν στις καταστάσεις των Things. Η σύνταξη και η δομή ενός rule file είναι κάπως έτσι:

```
//Import section
Import java.net.URI

//Variable Declaration section
var counter = 0
val msg = "This is a Message"
var Number x

//Rule section
rule "rule name"
when
     <trigger_condition1> or
     <trigger_condition2> or
     <trigger_condition3>
     …
then
     <script_block>
End
```

- **Services Files**

Τα services files βρίσκονται στον φάκελο /etc/openhab2/services και είναι τα αρχεία των εγκατεστημένων bindings. Σε αυτά τα αρχεία τροποποιούμε τα πεδία που μας ενδιαφέρουν έτσι όπως εμείς επιθυμούμε για την ορθή λειτουργεία του server. Παράδειγμα, όταν εγκαταστήσουμε το MQTT binding, μπορούμε να πάμε σε αυτό τον φάκελο και θα βρούμε τα αρχεία mqtt.cfg και mqtt-eventbus.cfg, όπου μπορούμε να ρυθμίσουμε τον MQTT server μας, δηλαδή σε ποια πόρτα ακούει, την IP διεύθυνση, το όνομα του broker κ.α.

3.5 Πρόσθετα (Add-ons)

Τα Add-ons είναι ένα κομμάτι εξίσου σημαντικό με τα configuration files για την διαμόρφωση του συστήματός μας, καθώς για κάθε συσκευή, πρωτόκολλο και υπηρεσία που θέλουμε να χρησιμοποιήσουμε πρέπει να προσθέσουμε στο σύστημά μας το αντίστοιχο Add-on. Τα Add-ons διακρίνονται στους εξής τύπους:

- **Bindings**
- **User Interfaces**
- **Persistence**
- **Actions**
- **Transformations**
- **Voice Services**
- **3rd Party System Integration**

Τα Bindings ενσωματώνουν φυσικό hardware, εξωτερικά συστήματα και web services στο openHAB. Υπάρχουν Bindings για τις περισσότερες smart συσκευές, αλλά και bindings όπως το MQTT binding και άλλων πρωτοκόλλων για την επίτευξη της επικοινωνίας μεταξύ του server και των μικροελεγκτών που έχουμε προγραμματίσει για την εξ' αποστάσεως λειτουργεία των συσκευών μας.

Το OpenHAB προσφέρει διάφορα User Interfaces όπως έχουμε προαναφέρει και από το πεδίο των πρόσθετων μπορούμε να κατεβάσουμε αυτά που χρειαζόμαστε. Οι διεπαφές αυτές είναι:

- **CLASSIC UI** – Διεπαφή για πρόσβαση στο Sitemap.
- **REST API** – Πρόσβαση στο Rest API.
- **BASIC UI** – Διεπαφή για πρόσβαση στο Sitemap (πιο μοντέρνα γραφικά).
- **PAPER UI** – Διεπαφή για την τροποποίηση/διαχείριση των Items και των Things, προσφέρει Add-ons management και σύνδεση καναλιών με Items.
- **HABMIN** – Διεπαφή που παρέχει λειτουργίες χρήστη και διαχείρισης.
- **HABPANEL** – Διεπαφή dashboard, ενδείκνυται για tablets και εντοιχισμένες συσκευές.
- **openHAB apps** – Android/iOS/Windows 10 εφαρμογές για την διαχείριση του συστήματος από smartphone και Windows PC, χωρίς την ανάγκη για browser.

Οι υπηρεσίες Persistence ενεργοποιούν την αποθήκευση των δεδομένων των καταστάσεων των Items με το πέρασμα του χρόνου. Μία τέτοια υπηρεσία είναι το MQTT persistence που μας επιτρέπει να τροφοδοτούμε την κατάσταση των Items σε ένα MQTT broker χρησιμοποιώντας τις persistence στρατηγικές του openHAB

Τα Actions είναι προκαθορισμένες μέθοδοι οι οποίες καλούνται από τους κανόνες και τα scripts του openHAB. Εισάγονται αυτόματα και μπορούν να χρησιμοποιηθούν για να εκτελέσουν λειτουργίες συγκεκριμένα για το openHAB ή να στείλουν εντολές ή δεδομένα σε εξωτερικές υπηρεσίες ή hardware. Υπάρχουν μερικά Actions που είναι προεγκατεστημένα στα συστήματα openHAB και Actions που είναι προαιρετικά, όπως το Mail Action που μας παρέχει SMTP υπηρεσίες έτσι ώστε οι κανόνες και τα script μας να μπορούν να στέλνουν e-mail. Σημαντικό action για σύστημα με μικροελεγκτές αποτελεί το MQTT Action που στέλνει μηνύματα σε topics κάποιου MQTT broker.

Τα transformation services όπως προαναφέραμε χρησιμοποιούνται για να μεταφράζουν δυσανάγνωστα δεδομένα σε πιο φιλική για τον χρήστη μορφή ή ακόμα και για την μετάφραση δεδομένων σε άλλη γλώσσα. Υπάρχουν αρκετά Transformation με τα πιο συνήθη να είναι το Map που λειτουργεί με τον τρόπο που προανέφερα στο υποκεφάλαιο με τα configuration files

και το JsonPath που εξάγει ένα στοιχείο ενός JSON string χρησιμοποιώντας ένα JsonPath expression.

Τέλος το openHAB πέραν των Voice services, όπως το MacOS Text – to – Speech υποστηρίζει υπηρεσίες που επιτρέπουν την ενσωμάτωση με διάφορες τεχνολογίες, παράδειγμα το Amazon Alexa Skill για την διαχείριση του συστήματος με φωνητικές εντολές μέσω του Amazon Echo Dot και το Dropbox όπου τα αρχεία του server μπορούν να συγχρονιστούν στον λογαριασμό dropbox του χρήστη για να κρατάει back – up.

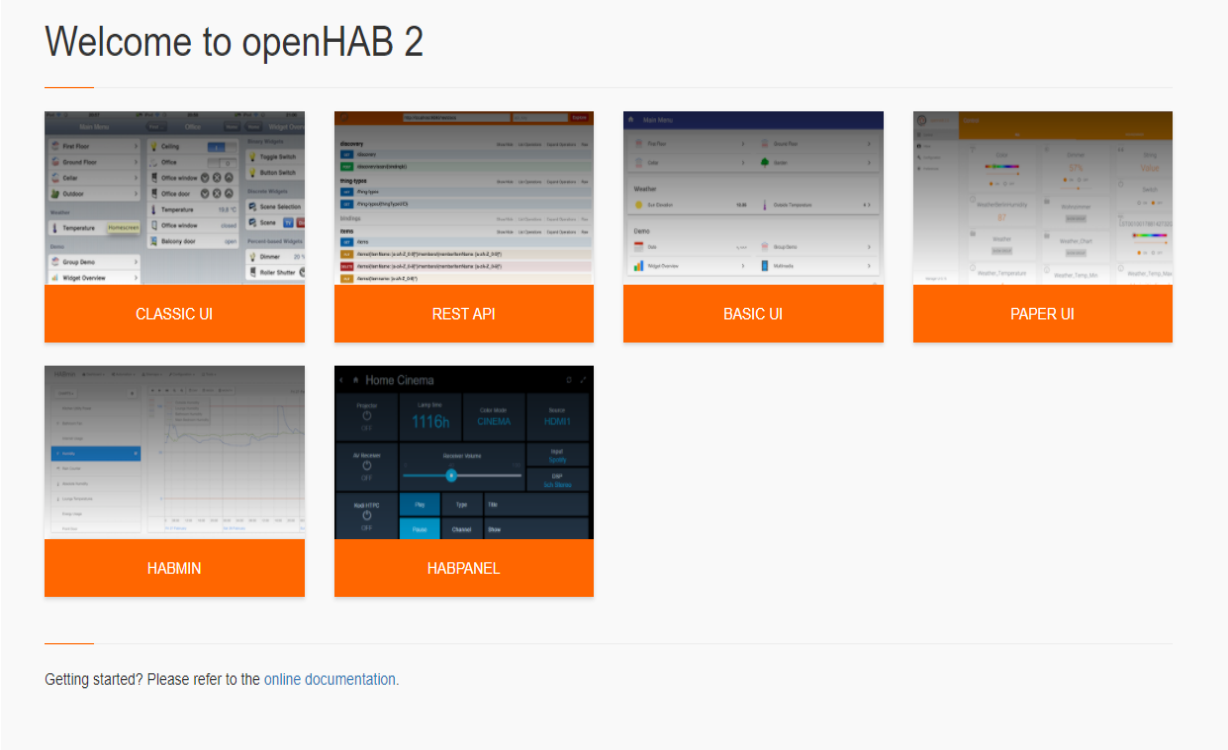

Figure 3-5 openHAB UI's

3.6 OpenHAB Cloud Connector

3.6.1 Σφαιρική εικόνα

Το openHAB cloud connector είναι μια υπηρεσία που επιτρέπει στον χρήστη να συνδέει το τοπικό δίκτυο που τρέχει το openHAB με απομακρυσμένο openHAB Cloud, όπως το myopenHAB.org, που αποτελεί μία openHAB cloud υπηρεσία που παρέχεται από το openHAB Foundation.

Το openHAB Cloud Service (άρα και το connector) είναι χρήσιμο για διάφορες περιπτώσεις όπως:

- Επιτρέπει την απομακρυσμένη πρόσβαση στο τοπικό δίκτυο του openHAB χωρίς χρειάζεται να εκθέσει ports στο διαδίκτυο ή την απαίτηση ρύθμισης πολύπλοκης VPN σύνδεσης.
- Αποτελεί συνδετικός κρίκος στο Google Cloud Messaging (GCM) και Apple Push Notifications (APN) για την αποστολή notifications στις smartphone εφαρμογές.
- Δίνει δυνατότητες ενσωμάτωσης υπηρεσιών που απαιτούν OAuth2 πιστοποίηση έναντι ενός web server, όπως το IFTTT ή το Amazon Alexa Skills.
- Επιτρέπει καταχώρηση επιπλέον απομακρυσμένων χρηστών για το σύστημα.

Το openHAB Cloud είναι κυρίως βασισμένο στα εξής frameworks και τεχνολογίες:

- Node.js – Server – side Javascript – framework.
- Express.js – Web application framework για το Node.js
- Nginx – Web server & reverse proxy
- MongoDB – NoSQL database
- redis – Session Manager & data structure server
- Socket.IO – Bi – directional communication μεταξύ web clients και servers

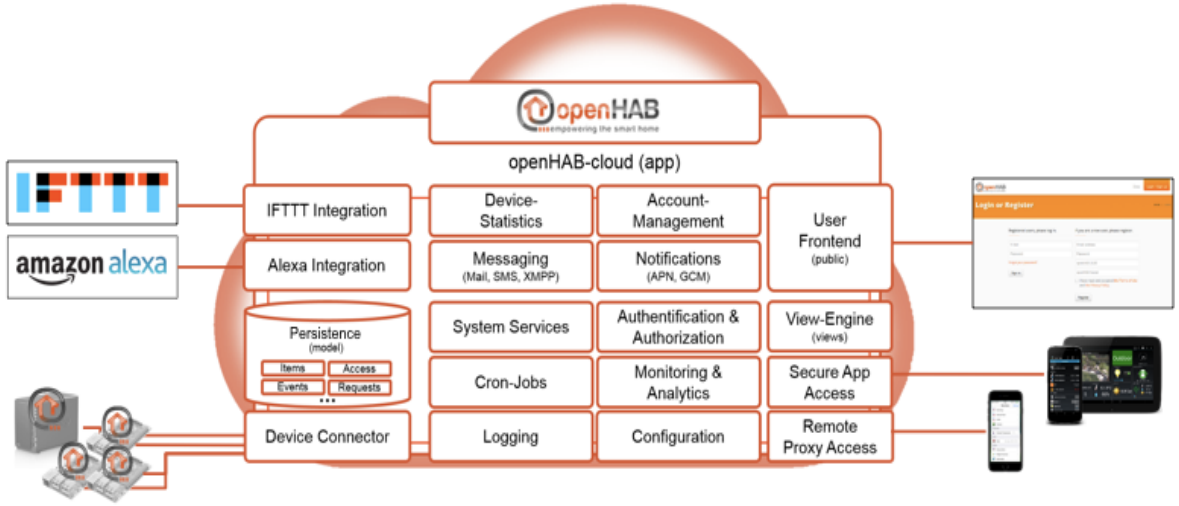

Figure 3-6 openHAB cloud connector

3.6.2 Πιστοποίηση

Για την πιστοποίηση με το openHAB Cloud, το τοπικό openHAB δημιουργεί δύο values, τα οποία πρέπει να δοθούν στις ρυθμίσεις λογαριασμού στο openHAB Cloud service. Το πρώτο είναι ένα μοναδικό identifier, που επιτρέπει να αναγνωρίσει openHAB runtime. Αποτελεί κάτι σαν username για την cloud πιστοποίηση. Το δεύτερο είναι ένα τυχαίο μυστικό κλειδί το οποίο εξυπηρετεί σαν κωδικός. Και τα δύο values είναι γραμμένα στο τοπικό file system. Αν χαθούν αυτά τα δύο για κάποιο λόγο, το openHAB θα δημιουργήσει 2 νέα αυτόματα. Σε αυτή την περίπτωση θα πρέπει να αλλάξουν και στις ρυθμίσεις λογαριασμού στο openHAB Cloud Service.

3.7 Android & iOS Applications

3.7.1 Σφαιρική Εικόνα

Το openHAB προσφέρει και εφαρμογές για iOS και Android που χρησιμοποιούν το REST API με σκοπό να κάνουν render τα sitemaps που έχει δημιουργήσει ο χρήστης. Τα χαρακτηριστικά που προσφέρουν οι εφαρμογές είναι τα εξής:

- Έλεγχος του openHAB και του openHAB Cloud
- Λήψη notifications από το openHAB Cloud
- Αλλαγή των Items μέσω NFC tags
- Αποστολή φωνητικών εντολών στο openHAB
- Ανακάλυψη συσκευών και προσθήκη αυτών σαν Items
- Υποστηρίζει εντοιχισμένα tablets

3.7.2 Σύνδεση με OpenHAB Server

Προκειμένου να επιτευχθεί η σύνδεση της εφαρμογής με τον server του συστήματός μας πρέπει να πάμε στις ρυθμίσεις της εφαρμογής και να καταχωρίσουμε την IP διεύθυνση του server για την διαχείριση των sitemaps εντός του δικτύου. Για την διαχείριση μέσω cloud πρέπει να δώσουμε την διεύθυνση του cloud instance (https://myopenhab.org) και επιπλέον να καταχωρίσουμε τα credentials του χρήστη (username & password).

4 Ανάλυση και Μεθοδολογία Υλοποίησης

Σε αυτό το κεφάλαιο θα παρουσιάσουμε και θα αναλύσουμε το hardware και το software που χρησιμοποιήθηκε για την ανάπτυξη του έξυπνου σπιτιού. Οι οντότητες του hardware ποικίλουν αλλά τα βασικά κομμάτια είναι οι ασύρματοι μικροελεγκτές. Οι μικροελεγκτές που χρησιμοποιούνται είναι οι ESP8266 και με βάση αυτούς υλοποιούνται τα απαραίτητα ηλεκτρονικά κυκλώματα με τα απαραίτητα ηλεκτρονικά στοιχεία.

Για την υλοποίηση χρησιμοποιήθηκε κινητό Android. Τα αποτελέσματα λειτουργούν αντίστοιχα και σε λειτουργικό σύστημα iOS καθώς η πλατφόρμα openHAB είναι συμβατή και στα προϊόντα της Apple όπως προαναφέραμε στο κεφάλαιο 3. Η προσέγγιση της διεπαφής επιτυγχάνεται και από προσωπικό ηλεκτρονικό υπολογιστή χωρίς το λειτουργικό σύστημα να αποτελεί πρόβλημα καθώς η διεπαφή φορτώνεται από τον browser.

4.1 Ανάλυση Συστήματος

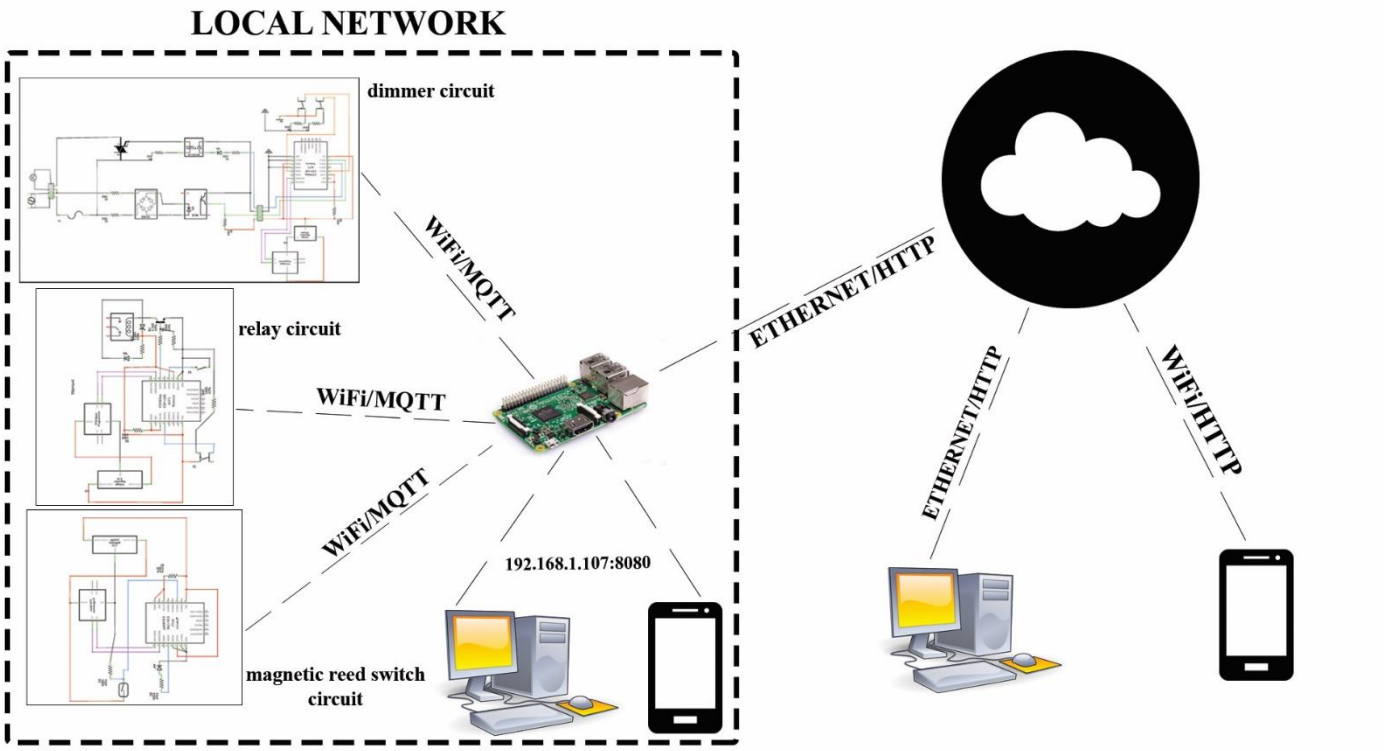

Figure 4-1 Home Automation Network

Στην εικόνα 4 – 1 περιγράφεται η αρχιτεκτονική του συστήματος που υλοποιούμε. Κεντρικός κόμβος του συστήματος είναι το Raspberry PI 3 που έχει τον ρόλο του εξυπηρετητή. Τα τερματικά του δικτύου είναι οι ασύρματοι μικροελεγκτές ESP8266 και είναι αυτοί οι οποίοι διαχειρίζονται τις συσκευές μέσω των ηλεκτρονικών κυκλωμάτων.

Μέσα στο τοπικό δίκτυο, ο χρήστης μπορεί με την χρήση των διεπαφών που προσφέρει το openHAB να διαχειριστεί τις συσκευές. Οι διεπαφές είναι προσεγγίσιμες στην πόρτα 8080 του server και η προσέγγιση γίνεται με την χρήση ενός browser ή την εφαρμογή του openHAB στο κινητό. Όταν ο χρήστης αλλάξει την κατάσταση της απεικόνισης της συσκευής στο openHAB, τότε ο εξυπηρετητής στέλνει το αντίστοιχο MQTT μήνυμα και ο μικροελεγκτής που διαχειρίζεται την αντίστοιχη συσκευή εκτελεί την τις εντολές που πρέπει. Η επικοινωνία των μικροελεγκτών με τον server γίνεται μέσω WiFi.

Για την διαχείριση του συστήματος εκτός δικτύου, χρησιμοποιείται η cloud υπηρεσία που προσφέρει το openHABκαι μέσω αυτής ο εξυπηρετητής ενημερώνεται για τυχόν αλλαγές στις καταστάσεις των αντικειμένων από τον χρήστη. Το Raspberry PI μπορεί να συνδεθεί στο διαδίκτυο είτε με WiFi είτε με ethernet. Στη συγκεκριμένη επιλογή, καθώς η θέση του server ήταν δίπλα στο router, χρησιμοποιήθηκε καλώδιο ethernet.

4.2 Ανάλυση Hardware

Οι μικροελεγκτές που χρησιμοποιήθηκαν για την υλοποίηση του δικτύου για τον έλεγχο των συσκευών είναι οι ESP8266 – 12F και ESP8266 – EVB. Πιο συγκεκριμένα το ESP8266 είναι ένα χαμηλού κόστους WiFi chip με ενσωματωμένο πρωτόκολλο TCP/IP και με δυνατότητα μικροελεγκτή, που παρασκευάζεται από την Espressif Systems. Επίσης χρησιμοποιήθηκε και ο μικροϋπολογιστής Raspberry PI 3 Model B σαν κεντρικός σαν διακομιστής και διάφορα ηλεκτρονικά στοιχεία για την υλοποίηση των κυκλωμάτων.

4.2.1 Raspberry PI 3 Model B

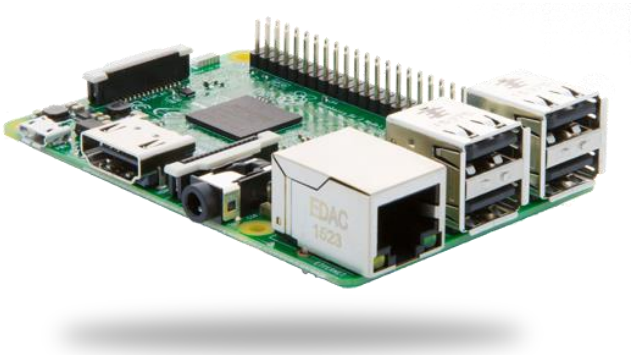

Figure 4-2 Raspberry PI 3 Model B

Το Raspberry PI 3 είναι ένας μικροϋπολογιστής τρίτης γενιάς Raspberry PI που κυκλοφόρησε τον Φεβρουάριο του 2016. Έχει την κανονική συμπεριφορά ενός υπολογιστή αλλά με αρκετά μικρότερες δυνατότητες από άποψη μνήμης και επεξεργαστικής ισχύς αλλά και γραφικών. Λειτουργεί με λειτουργικά συστήματα Linux με το πιο συνηθισμένο να είναι το Raspbian.

Στην δικιά μας εφαρμογή το Raspberry PI 3 αποτελεί το κεντρικότερο κομμάτι του συστήματος καθώς στην υλοποίηση χρησιμοποιείται σαν server του συστήματος και τρέχει το λειτουργικό σύστημα openHABian που είναι αντίστοιχο με το Raspbian με την διαφορά πως έχει ήδη εγκατεστημένα όλα τα πακέτα του openHAB τα οποία θα χρειαστούν για την διεκπεραίωση της εργασίας. Μέσα σε αυτόν τον μικροϋπολογιστή λοιπόν αποκτούν υπόσταση οι συσκευές και οι αισθητήρες του συστήματός μας από άποψη software και γίνεται όλη η διαχείριση των MQTT μηνυμάτων.

Τα χαρακτηριστικά που διέπουν αυτή την συσκευή είναι τα εξής:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40 – pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

4.2.2 ESP8266 – 12F

Figure 4-3 ESP8266 - 12F

Χαρακτηριστικά

- 802.11 b/g/n.
- Integrated low power 32 – bit MCU.
- Integrated 10 – bit ADC.
- Integrated TCP/IP protocol stack.
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators και power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval
- Deep sleep power < 10μA, Power down leakage current < 5μA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C
- FCC, CE, and ROSH certified

Παράμετροι

ΚΑΤΗΓΟΡΙΕΣ	ITEMS	VALUES
WiFi Parameters	Certificates	FCC/CE/ROSH
	WiFi Protocols	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Parameters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Control
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	16mm*24mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Software Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Table 3 ESP8266 - 12F Parameters

Περιγραφή των Pins

Το ESP8266 – 12F έχει συνολικά 18 pins για διάφορες χρήσεις, οι οποίες εξηγούνται στην εικόνα 4 – 4 και στον πίνακα 4.

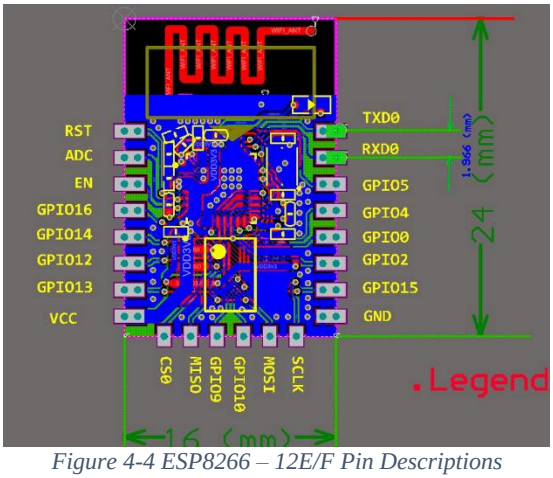

Figure 4-4 ESP8266 – 12E/F Pin Descriptions

NO	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result. Input voltage range 0-1v, scope:0-1024
3	EN	Chip enable pin. Active high
4	GPIO16	GPIO16; can be used to wake up the chipset from deep sleep mode
5	GPIO14	GPIO14; HSPI_CLK
6	GPIO12	GPIO12; HSPI_MISO
7	GPIO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input
11	IO9	GPIO9
12	IO10	GPIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	GPIO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	GPIO2	GPIO2; UART1_TXD
18	GPIO0	GPIO0
19	GPIO4	GPIO4
20	GPIO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Table 4 Pin Description

Ο μικροελεγκτής έχει επίσης 2 τρόπους λειτουργείας (pin modes) και μας είναι διαθέσιμοι με την γείωση / παροχή ρεύματος στις κατάλληλες GPIO. Ο ένας τρόπος λειτουργείας είναι το UART mode για τον προγραμματισμό του chip και ο άλλος είναι το Flash Boot mode για την λειτουργεία του chip.

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Table 5 Pin Mode

Περιγραφή Λειτουργειών

- **MCU:** Το ESP8266EX είναι ενσωματωμένο με έναν Tensilica L106 32 – bit μικροελεγκτή, ο οποίος έχει πολύ χαμηλή κατανάλωση ισχύος και 16 – bit RSIC. Το CPU clock είναι 80 MHz. Μπορεί επίσης να φτάσει την μέγιστη τιμή του η οποία είναι τα 160 MHz. Το ESP8266EX αρκετές φορές είναι ενσωματωμένο με διάφορους αισθητήρες και άλλες συγκεκριμένες συσκευές μέσω των GPIO του.

- **Οργάνωση Μνήμης:**
  - **Internal SRAM και ROM:** Το ESP8266EX είναι ενσωματωμένο με ελεγκτή μνήμης, συμπεριλαμβανομένης SRAM και ROM. Ο μικροελεγκτής μπορεί να επισκεφτεί τις μονάδες μνήμης μέσω iBus, dBus και AHB διεπαφές. Όλες οι μονάδες μνήμης είναι επισκέψιμες κατόπιν αιτήσεως, όσο ο «διαιτητής» μνήμης θα αποφασίζει την ακολουθία εκτέλεσης ανάλογα με το χρόνο λήψης αυτών των αιτημάτων από τον επεξεργαστή.
  - **External SPI Flash:** Το ESP8266 – 12F έχει 4 MB εξωτερική SPI Flash για να αποθηκεύει τα προγράμματα του χρήστη. Εάν απαιτήται μεγαλύτερος χώρος, τότε προτιμούμε ένα SPI Flash με μεγαλύτερη μνήμη. Θεωρητικά μπορεί να υποστηριχτεί χωρητικότητα μνήμης έως 16 MB.

Κατανάλωση Ενέργειας

Παράμετροι	Typical	Unit
Tx 802.11b, CCK 11Mbps, P OUT=+17dBm	170	mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm	140	mA
Tx 802.11n, MCS7, P OUT =+13dBm	120	mA
Rx 802.11b, 1024 bytes packet length , -80dBm	50	mA
Rx 802.11g, 1024 bytes packet length, -70dBm	56	mA
Rx 802.11n, 1024 bytes packet length, -65dBm	56	mA
Modem-Sleep[1]	15	mA
Light-Sleep[2]	0.9	mA
Deep-Sleep[3]	10	μA
Power Off	0.5	μA

Table 6 Κατανάλωση Ενέργειας

1 Modem Sleep: Απαιτεί τον επεξεργαστή να λειτουργεί, όπως στις εφαρμογές PWM ή I2S. Σύμφωνα με τα πρότυπα 802.11 (όπως το U – APSD), εξοικονομεί ενέργεια για να κλείσει το κύκλωμα του WiFi μόντεμ διατηρώντας παράλληλα μια σύνδεση WiFi χωρίς μετάδοση δεδομένων.

2 Light Sleep: η CPU μπορεί να ανασταλεί σε εφαρμογές όπως το Wi-Fi switch. Χωρίς μετάδοση δεδομένων. Το κύκλωμα του Wi-Fi μόντεμ μπορεί να απενεργοποιηθεί και η CPU αναστέλλεται για εξοικονόμηση ενέργειας σύμφωνα με το πρότυπο 802.11 (U-APSD).

3 Deep Sleep: δεν απαιτεί τη διατήρηση σύνδεσης Wi-Fi. Για εφαρμογή με μεγάλες καθυστερήσεις χρόνου μεταξύ των μεταδόσεων δεδομένων.

Σχηματικά

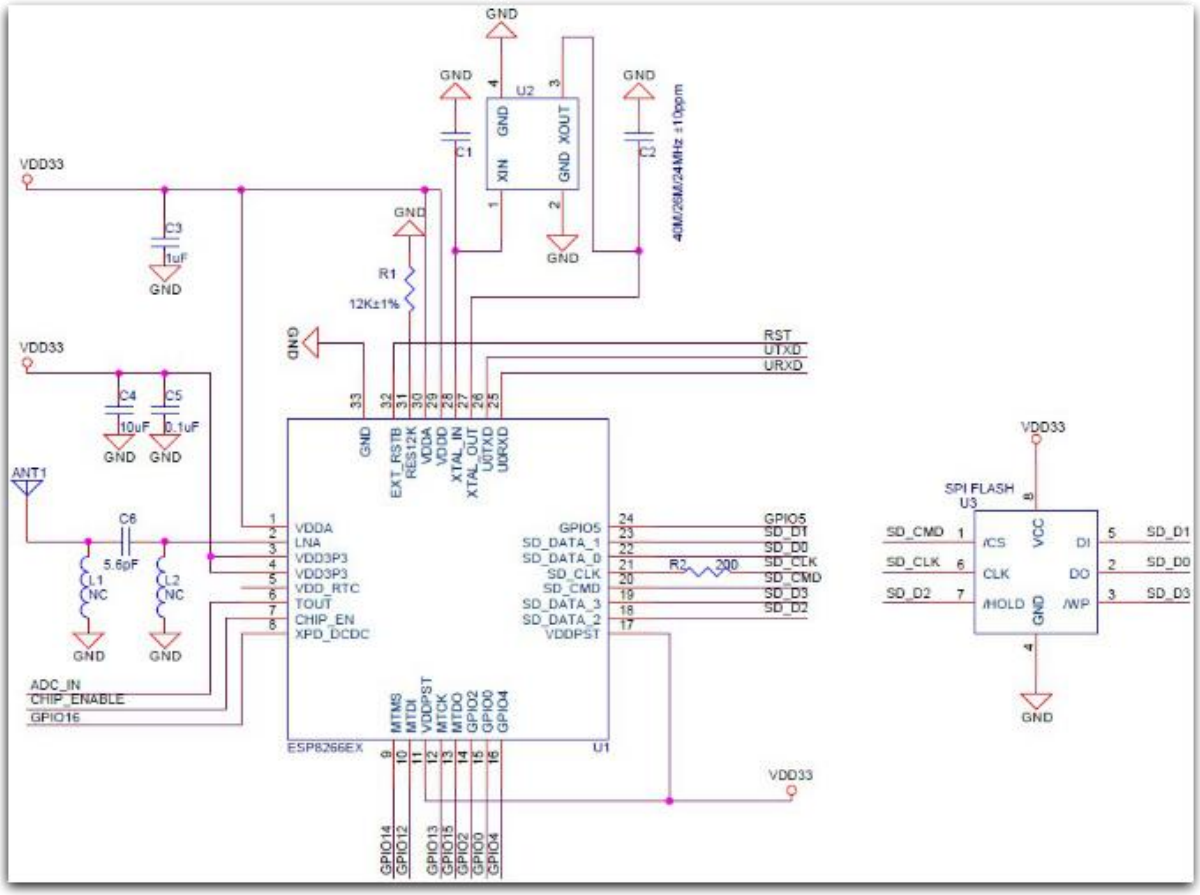

Figure 4-5 ESP8266 - 12F Schematics

4.2.3 ESP8266 – EVB

Το ESP8266 – EVB είναι μία πλακέτα της OLIMEX για το ESP8266EX με ρελέ, κουμπί, UEXT και όλες τις GPIO προσβάσιμες με ένα τυπικό header 0.1". Το κουμπί εξυπηρετεί για την εύκολη πρόσβαση στο UART mode για τον προγραμματισμό της πλακέτας. Η πλακέτα έχει πλήρως ενσωματωμένο το WiFi SoC ESP8266EX με όλες τις ιδιότητες και τα χαρακτηριστικά που προαναφέραμε.

Figure 4-6 ESP8266 – EVB

Χαρακτηριστικά Πλακέτας

- Περιλαμβάνει το MOD-WIFI-ESP8266-DEV.
- Ρελέ 10A/250VAC (15A/120VAC 15A/24VDC) with connector and status LED.
- Κουμπί για εύκολη πρόσβαση στο UART mode.
- Power jack για +5V τροφοδοσία.
- UEXT connector.
- Σειρά από 16 pin holes με πρόσβαση σε όλα τα pins του ESP8266.
- Διαστάσεις: (2.25 x 2)" ~ (57 x 50)mm.

Χαρακτηριστικά MOD – WiFi – ESP8266 – DEV

- Main chip: EPS8266EX.
- 2MB (16Mb) SPI flash memory.
- Power LED.
- User-programmable LED.
- SMT jumpers για διάφορα boot modes (FLASH, UART, SDO).
- PCB κεραία.
- UEXT pads για ευκολότερη πρόσβαση στο UART interface.
- Pads for a U.FL antenna connector (αν επιθυμούμε εξωτερική κεραία).
- 22 pin holes για ευκολότερη πρόσβαση στα pins του επεξεργαστή.
- OSHW design.
- Διαστάσεις: (1.3 x 0.9)" ~ (33 x 23)mm.

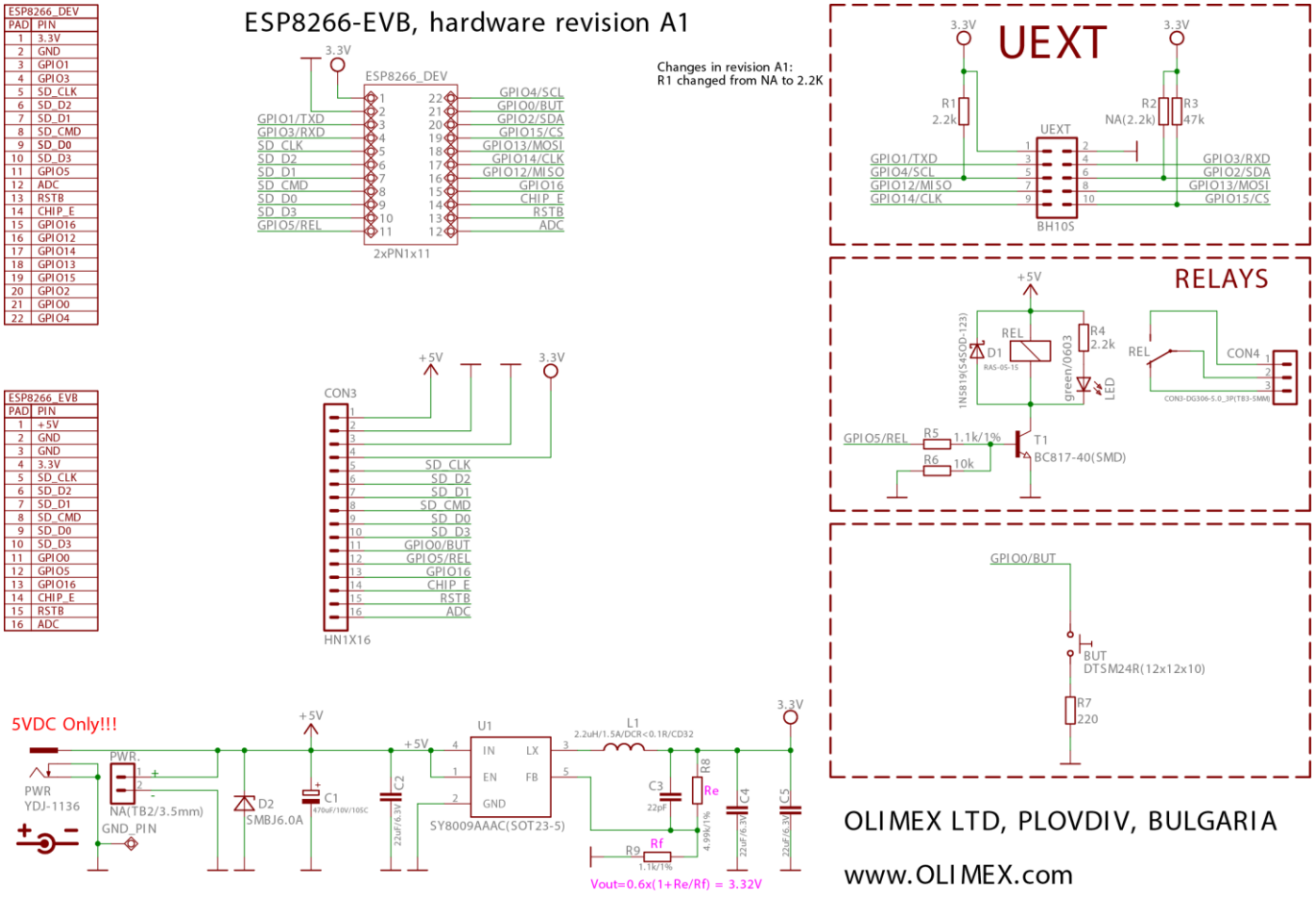

Figure 4-7 ESP8266 - EVB datasheet

4.2.4 Ηλεκτρονικά Στοιχεία

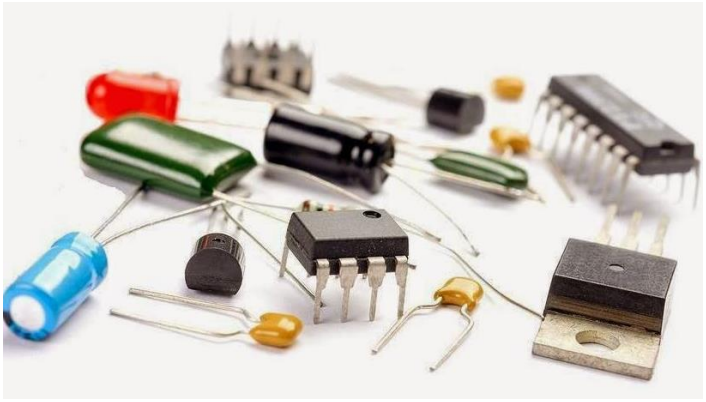

Figure 4-8 Ηλεκτρονικά Στοιχεία

TRIAC

Το TRIAC αποτελεί ακρωνύμιο του triode for alternative current. Είναι ένα ηλεκτρονικό στοιχείο με τρία τερματικά που άγει ρεύμα προς οποιαδήποτε κατεύθηνση όταν πυροδοτείται. Το επίσημο όνομα είναι διπλή κατεύθυνσης τριοδικό θυρίστορ ή διμερές / αμφίπλευρο τριοδικό θυρίστορ. Ένα θυρίστορ αποτελεί ανάλογο του ρελέ καθώς με μία μικρή τάση και ένταση ρεύματος μπορεί να ελέγχει ένα ρεύμα μεγαλύτερης τάσης και έντασης. Στην εφαρμογή μας χρησιμοποιούμε ένα TRIAC TIC – 206m στο κύκλωμα του dimmer.

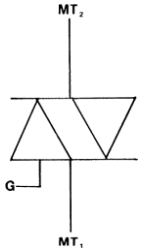

Figure 4-9 TRIAC TIC - 206m

Οπτοσυζευκτές

Οι οπτοσυζευκτές (optocouplers / opto – isolators / photocouplers) είναι ηλεκτρονικά στοιχεία που μεταφέρουν ηλεκτρικά σήματα μεταξύ 2 απομονωμένων κυκλωμάτων χρησιμοποιώντας φως. Ο λόγος είναι για να προληφθεί τυχών ζημιά στο κύκλωμα που λαμβάνει το σήμα από την μεγάλη τάση του άλλου κυκλώματος. Στην εφαρμογή μας χρησιμοποιούμε ένα optocoupler 4n35 και ένα MOC3021 στο κύκλωμα του dimmer, το οποίο είναι πιο συγκεκριμένα optotriac.

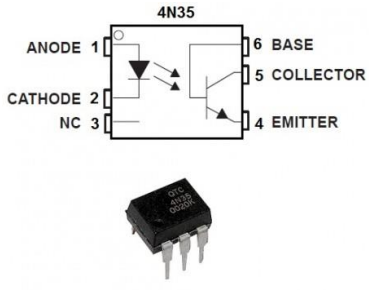

Figure 4-10 optocoupler 4n35

Ανορθωτής Γέφυρας

Ο ανορθωτής γέφυρας είναι ένα κύκλωμα που αποτελείται από τέσσερις διόδους και χρησιμοποιείται σε κυκλώματα μετατροπής του εναλλασσόμενου ρεύματος σε συνεχές. Αυτό που κάνει είναι να επιτρέπει μόνο τον θετικό παλμό να περνάει στο κύκλωμα αν είναι ημιανορθωτής ή να μετατρέπει τον αρνητικό παλμό σε θετικό αν είναι πλήρης ανορθωτής. Στο κύκλωμα του dimmer χρησιμοποιούμαι ένα πλήρη ανορθωτή γέφυρας KBP04, 400V.

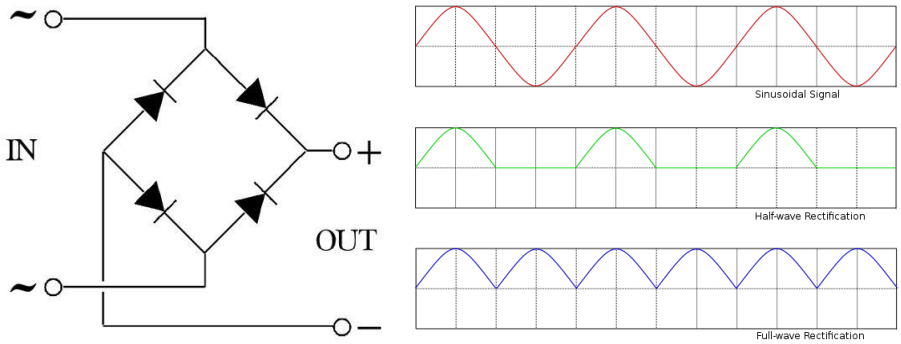

Figure 4-11 Full Bridge Rectifier / Half-rectifier & Full-rectifier Output

Ασφάλεια

Η ασφάλεια είναι μία συσκευή που παρέχει προστασία στο κύκλωμα για να μην περάσει υπερβολικό ρεύμα σε περίπτωση υπέρτασης. Είναι ένα βασικό εξάρτημα που αποτελείται από μία λεπτή μεταλλική χορδή η οποία λιώνει όταν τη διαπερνά υπερβολικό ρεύμα με αποτέλεσμα να μην περνάει στο υπόλοιπο κύκλωμα καταστρέφοντάς το. Στην περίπτωση αυτή, η ασφάλεια πρέπει μετά να αντικατασταθεί ή να αλλαχθεί το καλώδιο· τι από τα δύο θα γίνει εξαρτάται από τον τύπο της ασφάλειας.

Figure 4-12 Ηλεκτρική Ασφάλεια

Μαγνητικός Διακόπτης

Ο μαγνητικός διακόπτης (reed switch) είναι ένα διακόπτης που λειτουργεί με την εφαρμογή μαγνητικού πεδίου. Αποτελείται από δύο επαφές. Αυτές οι επαφές δεν αγγίζουν μεταξύ τους παρά μόνο αν υπάρξει κάποιο μαγνητικό πεδίο που θα τις φέρει σε επαφή με αποτέλεσμα να περάσει το ρεύμα. Ένα τέτοιο διακόπτη χρησιμοποιούμε στο κύκλωμα ασφαλείας στον οικιακό αυτοματισμό στις πόρτες του σπιτιού. Ο διακόπτης που χρησιμοποιούμαι συγκεκριμένα είναι normally close (NC), δηλαδή αγγίζουν οι επαφές χωρίς το μαγνητικό πεδίο και με το μαγνητικό πεδίο απομακρύνονται. Ο μαγνήτης είναι πάνω στην πόρτα και όταν η πόρτα ανοίξει και το κύκλωμα κλείσει τότε στέλνεται στον μικροελεγκτή σήμα και αυτός με την σειρά του κάνει ό,τι χρειάζεται να κάνει.

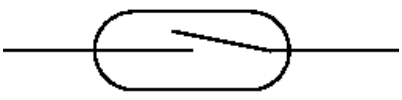

Figure 4-13 Reed Switch

Ρυθμιστής Τάσης

Ο ρυθμιστής τάσης (voltage regulator) είναι μία συσκευή η οποία παρέχει στο κύκλωμα την επιθυμητή τάση. Για παράδειγμα οι μικροελεγκτές ESP8266 – 12F έχουν τάση λειτουργείας τα 3.3 Volt αλλά οι διαθέσιμη μπαταρία είναι στα 9 Volt και η τροφοδοσία από το USB του ηλεκτρονικού υπολογιστή είναι στα 5 Volt. Ο ρυθμιστής παίρνει αυτή την τάση και την ρίχνει στα 3.3 Volt για να λειτουργήσει το κύκλωμα χωρίς να υπάρχει κίνδυνος να καεί ο μικροελεγκτής.

Αντιστάσεις

Η αντίσταση είναι ένα παθητικό ηλεκτρονικό στοιχείο με δύο τερματικά που παρέχει ηλεκτρική αντίσταση όπου χρειάζεται στο κύκλωμά μας. Χρησιμοποιούνται στα κυκλώματα για να μειωθεί η ροή ρεύματος, να προσαρμόσουν τα επίπεδα του σήματος, να διαιρέσουν τάσεις και να τερματίσουν γραμμές μεταφοράς κ.α.

4.2.5 Smartphone

Τα smartphones τα τελευταία χρόνια έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας όλων των ανθρώπων καθώς προσφέρουν μία δυνατή υπολογιστική ισχύ και είναι εξαιρετικά ευέλικτα και εύκολα στην μεταφορά τους. Το γεγονός αυτό, τα έχει καταστήσει να αποτελούν ένα βασικό κομμάτι μέσα στον κόσμο του Internet of Things καθώς μπορείς να έχεις όλες τις όψεις ενός συστήματος στο χέρι σου και η αλληλεπίδραση με αυτό είναι εξαιρετικά εύκολη και γρήγορη.

Για αυτό τον λόγο κάθε οικιακός αυτοματισμός που αναπτύσσεται έχει σαν κομμάτι του και το smartphone καθώς αυτό μας δίνει την δυνατότητα να παρακολουθούμε και να αλληλεπιδράμε με το σύστημα οπουδήποτε και αν είμαστε στον κόσμο μέσω των cloud υπηρεσιών.

Έτσι και στην δικιά μας εφαρμογή η επίδραση με το σύστημα του οικιακού αυτοματισμού επιτυγχάνεται κυρίως με την χρήση του smartphone μέσω των διεπαφών που παρέχει το openHAB για Android συστήματα αλλά και για iOS.

4.2.6 Ηλεκτρονικός Υπολογιστής

Ο ηλεκτρονικός υπολογιστής, όπως και το smartphone αποτελούν μεγάλο κομμάτι της καθημερινότητας του ανθρώπου και αποτελεί επίσης βασικό εργαλείο για την χρήση ενός συστήματος Internet of Things, πόσο μάλλον και για την χρήση ενός συστήματος οικιακού αυτοματισμού.

Πέραν της χρήσης του για την διαχείριση ενός συστήματος Internet of Things αποτελεί το βασικό εργαλείο για την ανάπτυξη του συστήματος καθώς στην παρούσα εργασία μέσω του ηλεκτρονικού υπολογιστή προγραμματιστήκανε όλοι οι μικροελεγκτές, στήθηκε ο server με την χρήση Secure Shell (SSH) για την απομακρυσμένη διαχείριση και χρήση του server μέσα στο τοπικό δίκτυο, δοκιμάστηκαν οι επικοινωνίες με την χρήση του MQTT πρωτοκόλλου για την διαπίστωση της ορθής λειτουργείας και επιπλέον σχεδιαστήκαν και τα κυκλώματα των μικροελεγκτών με τους αισθητήρες και τις συσκευές.

Για την δημιουργία και τον προγραμματισμό ενός τέτοιου συστήματος, οποιοσδήποτε ηλεκτρονικός υπολογιστής που το λειτουργικό του σύστημα είναι συμβατό με την χρήση του κατάλληλου software, είναι ικανός να ανταπεξέλθει. Για την δημιουργία του συγκεκριμένου συστήματος χρησιμοποιήθηκε ένα υπολογιστής με λειτουργικό σύστημα Windows 10, μνήμη RAM 8 GB και επεξεργαστή intel i5 6600k στα 3.5 GHz.

4.3 Ανάλυση Software

Σε αυτό το μέρος αναφέρουμε και αναλύουμε ό,τι είδους software χρησιμοποιήθηκε για την υλοποίηση του συστήματος οικιακού αυτοματισμού. Το λειτουργικό σύστημα του Raspberry PI 3, openHABian, τον προγραμματισμό των μικροελεγκτών και τα εργαλεία και τις βιβλιοθήκες που χρησιμοποιήθηκαν, την δοκιμή των MQTT επικοινωνιών, την SSH διαχείριση και χρήση του server αλλά και διάφορες μεθόδους που χρησιμοποιήθηκαν για την υλοποίηση απομακρυσμένου προγραμματισμού των μικροελεγκτών αλλά και αλγόριθμους για την αντιμετώπιση φυσικών προβλημάτων όπως το bouncing.

4.3.1 openHABian

Το openHAB 2 όπως προαναφέραμε στο κεφάλαιο 3, είναι γραμμένο σε Java και βασισμένο στο Eclipse SmartHome framework. Οπότε χρειάζεται μόνο μία Java Virtual Machine, η οποία είναι διαθέσιμη για πολλές πλατφόρμες με αποτέλεσμα το openHAB να μπορεί να εκτελεστεί από διάφορες εκδόσεις των Mac OS X και Windows και διάφορες παραλλαγές των Linux (Ubuntu, Raspbian, …).

Παρ’ όλα αυτά, η χρήση του openHAB σε μικροϋπολογιστές, όπως το Raspberry Pi, είναι ευρέως διαδεδομένη, αλλά και πιο πρακτική, πράγμα που καθιστά την χρήση Linux αναπόφευκτη. Καθώς όμως η πλήρης ρύθμιση ενός λειτουργικού συστήματος Linux μαζί με όλα τα πακέτα και τις συστάσεις του openHAB παίρνει πολύ ώρα και αφήνει περιθώρια λάθους, ενδείκνυται η χρήση του openHABian. Το openHABian αποτελεί ένα self-configuring Linux σύστημα για SD cards, που περιέχει όλα τα πακέτα που μπορεί κάποιος χρήστης να χρειαστεί κατά την διάρκεια της υλοποίησης ενός οικιακού αυτοματισμού.

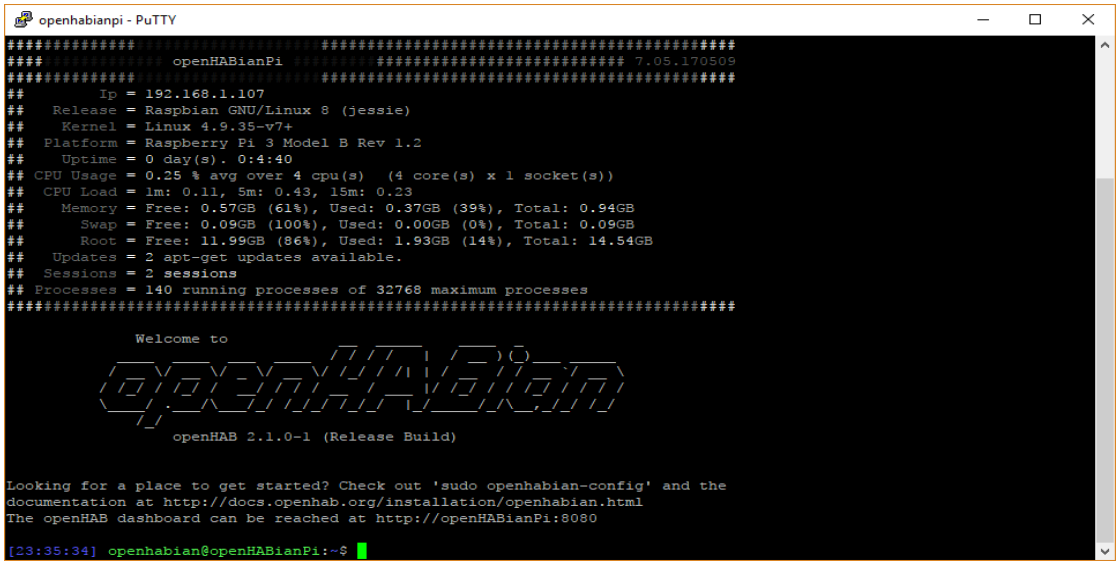

Figure 4-14 openHABian

Τέλος, η διαμόρφωση βασικών σημείων του λειτουργικού συστήματος openhabian γίνεται με την χρήση του εργαλείου openhab-config. Περισσότερες λεπτομέρειες γι’ αυτό θα δούμε στο κεφάλαιο 4.

4.3.2 Secure Shell & PuTTY

Το Secure Shell (SSH) είναι ένα κρυπτογραφικό πρωτόκολλο δικτύου για την ασφαλή λειτουργία των δικτυακών υπηρεσιών πάνω σε ένα μη ασφαλές δικτύου. Η πιο γνωστή εφαρμογή είναι η απομακρυσμένη σύνδεση στα συστήματα υπολογιστών από τους χρήστες.

Το SSH παρέχει ένα ασφαλές κανάλι επικοινωνίας πάνω σε ένα μη ασφαλές δίκτυο αρχιτεκτονικής client – server, συνδέοντας μία SSH εφαρμογή – client με ένα SSH – server. Συνηθισμένη είναι η χρήση του πρωτοκόλλου για απομακρυσμένο log – in μέσω γραμμής εντολών και απομακρυσμένη εκτέλεση εντολών. Κάθε δίκτυο μπορεί να ασφαλιστεί με SSH. Το πρωτόκολλο διακρίνεται σε δύο βασικές εκδόσεις, την SSH – 1 και SSH – 2.

Για την διαχείριση του server του συστήματος (raspberry PI 3), λοιπόν, χρησιμοποιήθηκες το PuTTY. Το PuTTY είναι ένα πρόγραμμα για SSH sessions κατάλληλο για χρήστες Windows καθώς τα Windows δεν παρέχουν από μόνα τους την δυνατότητα επικοινωνίας με SSH όπως τα Macintosh για παράδειγμα. Για την επικοινωνία με τον server μας απλά συμπληρώνουμε την IP του server, την πόρτα επικοινωνίας και τον τύπο της σύνδεσης (SSH προφανώς).

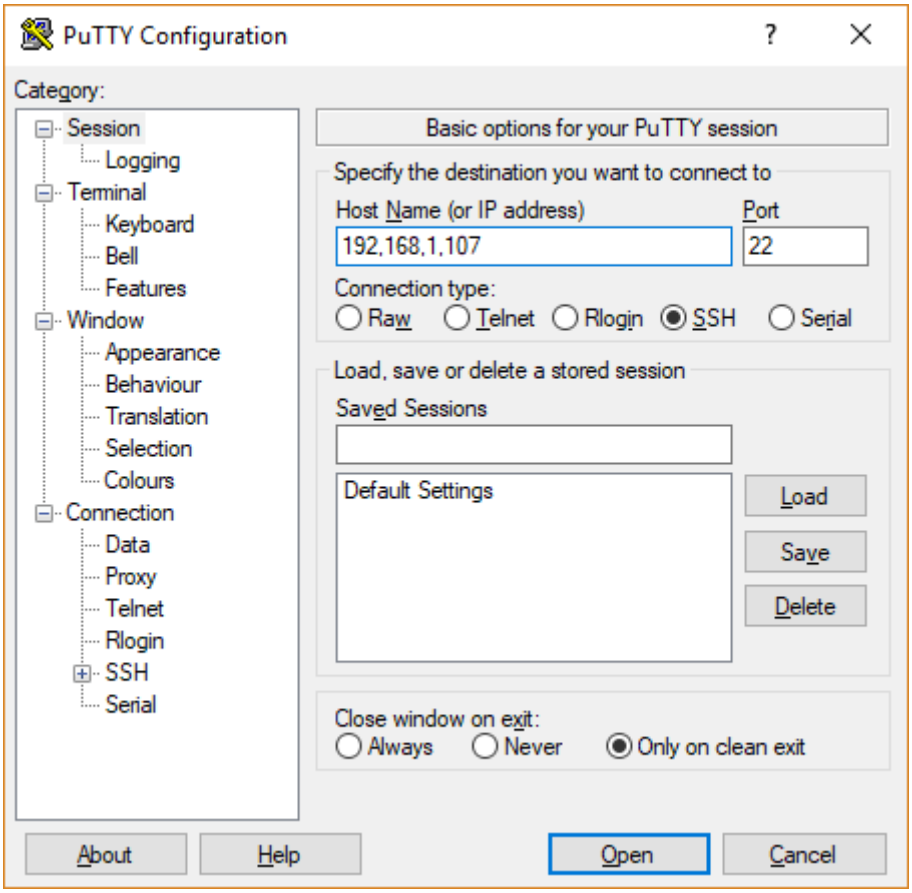

Figure 4-15 PuTTY SSH connection

4.3.3 Arduino IDE

Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino είναι μία εφαρμογή γραμμένη σε Java, που λειτουργεί σε πολλές πλατφόρμες και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Έχει σχεδιαστεί για να εισαγάγει τον προγραμματισμό στους καλλιτέχνες και τους νέους που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα με χαρακτηριστικά όπως είναι η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και είναι επίσης σε θέση να

μεταγλωττίζει και να φορτώνει προγράμματα στην πλακέτα με ένα μόνο κλικ. Δεν υπάρχει συνήθως καμία ανάγκη να επεξεργαστείτε αρχεία make ή να τρέξετε προγράμματα σε ένα περιβάλλον γραμμής εντολών. Ένα πρόγραμμα ή κώδικας που γράφτηκε για Arduino ονομάζεται σκίτσο (sketch).

Τα Arduino προγράμματα είναι γραμμένα σε C ή C++. Το Arduino IDE έρχεται με μια βιβλιοθήκη λογισμικού που ονομάζεται "Wiring", από το πρωτότυπο σχέδιο Wiring, γεγονός που καθιστά πολλές κοινές λειτουργίες εισόδου/εξόδου πολύ πιο εύκολες. Οι χρήστες πρέπει μόνο να ορίσουν δύο λειτουργίες για να κάνουν ένα πρόγραμμα κυκλικής εκτέλεσης:

- -setup():μία συνάρτηση που τρέχει μία φορά στην αρχή του προγράμματος η οποία αρχικοποιεί τις ρυθμίσεις
- -loop():μία συνάρτηση που καλείται συνέχεια μέχρι η πλακέτα να απενεργοποιηθεί

Ένα τυπικό πρώτο πρόγραμμα για έναν μικροελεγκτή αναβοσβήνει απλά ένα LED. Στο περιβάλλον του Arduino, ο χρήστης μπορεί να γράψει ένα πρόγραμμα σαν αυτό:

```
 #define LED_PIN 13

void setup () {
 pinMode (LED_PIN, OUTPUT); // enable pin 13 for digital
output
}

void loop () {
 digitalWrite (LED_PIN, HIGH); // turn on the LED
 delay (1000); // wait one second (1000 milliseconds)
 digitalWrite (LED_PIN, LOW); // turn off the LED
 delay (1000); // wait one second
}
```

Είναι ένα χαρακτηριστικό των περισσότερων πλακετών Arduino ότι έχουν ένα LED και μία αντίσταση φορτίου που συνδέονται μεταξύ του pin 13 και του εδάφους, ένα βολικό χαρακτηριστικό για πολλά απλά τεστ. Ο προηγούμενος κώδικας δεν θα αναγνωριστεί από ένα κανονικό μεταγλωττιστή C + + ως έγκυρο πρόγραμμα, έτσι ώστε όταν ο χρήστης κάνει κλικ στο κουμπί "Upload to I / O board" στο IDE, ένα αντίγραφο του κώδικα θα γραφτεί σε ένα προσωρινό αρχείο με ένα παραπάνω include στην κορυφή και μία πολύ απλή συνάρτηση main() στο τέλος, για να φτιάξει ένα έγκυρο C++ πρόγραμμα.

Το IDE του Arduino χρησιμοποιεί το GNU toolchain και το AVR Libc για να μεταγλωττίζει προγράμματα και το avrdude για να φορτώνει προγράμματα στην πλακέτα.

Δεδομένου ότι η πλατφόρμα Arduino χρησιμοποιεί Atmel μικροελεγκτές, το περιβάλλον ανάπτυξης της Atmel, το AVR Studio ή το νεότερη έκδοση του Atmel Studio, μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη λογισμικού για το Arduino.

Πέραν όμως της χρήσης του IDE για Arduino Boards μπορούμε να το χρησιμοποιήσουμε για να φορτώσουμε sketches και σε επιπλέον μικροελεγχτές όπως το ESP8266 που μας ενδιαφέρει για την παρούσα εργασία.

4.3.4 Over the Air (OTA) Programming

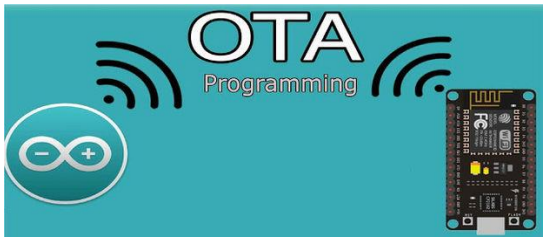

Figure 4-16 OTA programming

Σφαιρική Εικόνα

Το OTA αναφέρεται σε διάφορες μεθόδους για την διανομή νέου λογισμικού, διαμορφώσεις ρυθμίσεων και ακόμα ενημέρωση των κρυπτογραφικών κλειδιών σε συσκευές όπως smartphones κ.α. Ένα σημαντικό χαρακτηριστικό του OTA είναι ότι μία κεντρική τοποθεσία μπορεί να στείλει μία ενημέρωση σε όλους τους χρήστες, που δεν έχουν την ικανότητα να αρνηθούν ή να αλλάξουν την ενημέρωση, και αυτή η ενημέρωση εφαρμόζεται άμεσα σε όλους που είναι συνδεδεμένοι στο κανάλι.

Ο μηχανισμός OTA απαιτεί το υπάρχον λογισμικό και το υλικό της συσκευής προορισμού να υποστηρίζει τη λειτουργία, δηλαδή την παραλαβή και την εγκατάσταση νέου λογισμικού που λαμβάνεται μέσω του ασύρματου δικτύου από τον πάροχο. Νέο λογισμικό μεταφέρεται στο τηλέφωνο, εγκαθίσταται και τίθεται σε λειτουργία. Είναι συχνά απαραίτητη η επανεκκίνηση του τηλεφώνου για να εφαρμοστεί η νέα ενημέρωση.

OTA update σε ESP8266

Σύμφωνα με τα παραπάνω, το OTA αποτελεί την μέθοδο αποστολής νέου firmware στο ESP module με την χρήση WiFi σύνδεσης και όχι της σειριακής θύρας. Αυτή η λειτουργικότητα είναι εξαιρετικά χρήσιμη για περιπτώσεις στις οποίες η φυσική πρόσβαση στην συσκευή είναι από περιορισμένη μέχρι και αδύνατη.

Το OTA μπορεί να πραγματοποιηθεί με την χρήση:

- Arduino IDE
- Web Browser
- HTTP server

Η επιλογή Arduino IDE προορίζεται κυρίως για τη φάση ανάπτυξης λογισμικού. Οι άλλες δύο επιλογές είναι πιο χρήσιμες μετά την ανάπτυξη, για την παροχή στον μικροελεγχτή ενημερώσεις εφαρμογών με ένα web browser ή αυτόματα με την χρήση ενός http server. Σε κάθε περίπτωση, η πρώτη φόρτωση λογισμικού στον μικροελεγχτή θα πρέπει να γίνει με την σειριακή θύρα και αν το OTA εφαρμοστεί σωστά τότε όλα τα υπόλοιπα uploads μπορούν να γίνονται μέσω WiFi.

Το OTA δεν υπαγορεύει κάποια διαδικασία ασφάλειας. Εξαρτάται από τον προγραμματιστή η εξασφάλιση ότι οι ενημερώσεις επιτρέπονται μόνο σε θεμιτές πηγές. Μόλις η ενημέρωση ολοκληρωθεί, ο μικροελεγχτής κάνει επανεκκίνηση και ο νέος κώδικας εκτελείται. Ο προγραμματιστής πρέπει να εξασφαλίσει η εφαρμογή που τρέχει ο μικροελεγχτής θα κλείσει και θα ξανανοίξει με ασφάλεια.

Ασφάλεια

Ο μικροελεγχτής θα πρέπει να εκτεθεί ασύρματα για να λάβει τις νέες ενημερώσεις. Αυτό αυξάνει τις πιθανότητες για να τον παραβιάσουν και να φορτωθεί κάποιο ανεπιθύμητο πρόγραμμα. Για να μειωθούν οι πιθανότητες του hacking μπορούμε να προστατέψουμε τον μικροελεγχτή με κωδικό, διαλέγοντας συγκεκριμένη θύρα κ.α. Για παράδειγμα μπορούμε να

χρησιμοποιήσουμε στο Arduino IDE την βιβλιοθήκη Arduino OTA που διαθέτει τέτοιου είδους μεθόδους.

```
void setPort(uint16_t port);
void setHostname(const char* hostname);
void setPassword(const char* password);
```

Σιγουριά

Η διαδικασία του OTA καταλαμβάνει αρκετούς πόρους και εύρος ζώνης κατά την διάρκεια της φόρτωσης της ενημέρωσης. Έπειτα ο μικροελεγχτής επανεκκινείται και το sketch εκτελείται. Τώρα πρέπει να αναλύσουμε και δοκιμάσουμε πως επιδρά στην λειτουργικότητα του συστήματος με το νέο και το παλιό sketch.

Αν το ESP8266 είναι τοποθετημένο σε μία απομακρυσμένη τοποθεσία και ελέγχει κάποιον εξοπλισμό, θα πρέπει να προσέξουμε ιδιαιτέρως τι συμβαίνει στον εξοπλισμό εάν διακοπεί η λειτουργεία του απότομα για την διαδικασία της ενημέρωσης. Επομένως πρέπει να αποφασίσουμε πως θα φέρουμε τον εξοπλισμό σε μία ασφαλή κατάσταση πριν ξεκινήσουμε την ενημέρωση. Για παράδειγμα το ESP8266 μπορεί να ελέγχει ένα σύστημα ποτίσματος σύμφωνα με ένα χρονοδιάγραμμα. Εάν δεν προσέξουμε και μείνει κάποια βαλβίδα του εξοπλισμού ανοιχτή ο κήπος μπορεί να πλημυρίσει αν δεν κλείσει η βαλβίδα μετά την ενημέρωση και επανεκκίνηση του μικροελεγκτή.

Οι ακόλουθες συναρτήσεις παρέχονται επίσης από την βιβλιοθήκη OTA και προορίζονται για την διαχείριση της εφαρμογής κατά την διάρκεια συγκεκριμένων σταδίων του OTA ή σφάλματος του OTA.

```
void onStart(OTA_CALLBACK(fn));
void onEnd(OTA_CALLBACK(fn));
void onProgress(OTA_CALLBACK_PROGRESS(fn));
void onError(OTA_CALLBACK_ERROR (fn));
```

Βασικές προϋποθέσεις για την χρήση του OTA είναι ότι η μνήμη flash πρέπει να είναι τουλάχιστον διπλάσια από το μέγεθος του sketch και ότι ο υπολογιστείς που θα στείλει την ενημέρωση καθώς και ο μικροελεγκτής πρέπει να βρίσκονται μέσα στο ίδιο τοπικό δίκτυο.

Για να μπορούμε να χρησιμοποιήσουμε το OTA πρέπει πρώτα να κάνουμε κάποιες διαδικασίες. Κατ' αρχήν να εγκαταστήσουμε την Python 2.7 (ή νεότερη έκδοση) στον τοπικό σκληρό δίσκο. Έπειτα πάμε στις ιδιότητες του συστήματός μας και επιλέγουμε τις μεταβλητές περιβάλλοντος και προσθέτουμε μία νέα με όνομα PY_HOME και τιμή μεταβλητής C:\Python27\Lib;C:\Python27\DLLs;C:\Python27\Lib\lib-tk;C:\Python27;. Έπειτα κάνουμε επανεκκίνηση στο σύστημά μας.

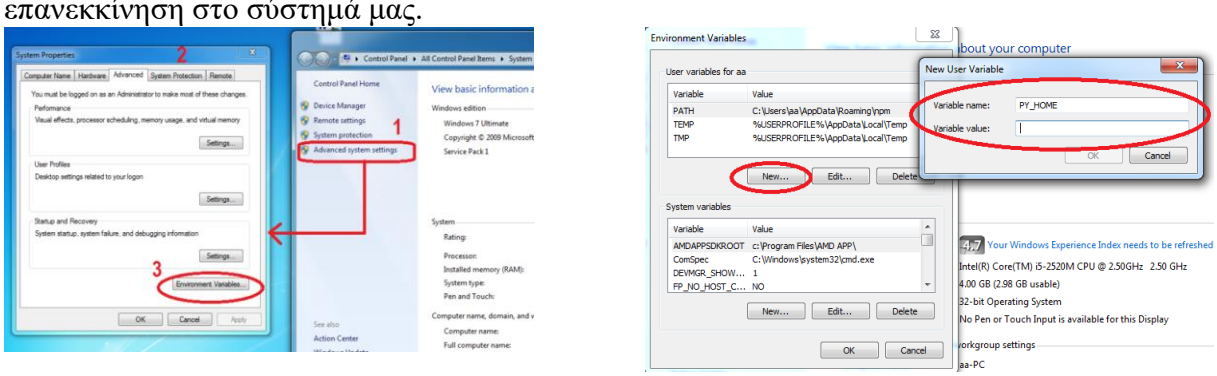

Figure 4-17 Python Variable

Αφού γράψουμε τον κώδικα και τον κάνουμε upload με τον συμβατικό τρόπο (USB) πρέπει να κάνουμε επανεκκίνηση του Arduino IDE και μετά την μεταγλώττιση του προγράμματος μπορούμε να δούμε στις θύρες μία IP διεύθυνση, η οποία ανήκει στον μικροελεγκτή μας. Την επιλέγουμε σαν θύρα upload και πλέον ενημερώνουμε τον μικροελεγκτή μας από το WiFi με την χρήση του Arduino IDE.

Figure 4-18 OTA port

4.3.5 MQTTfx

Figure 4-19 MQTTfx logo

Το MQTTfx είναι ένα πρόγραμμα με το οποίο συνδεόμαστε στον mqtt server που μας ενδιαφέρει καταχωρώντας στις ρυθμίσεις σύνδεσης την IP διεύθυνσή του. Αφού συνδεθούμε μπορούμε να κάνουμε εγγραφή στα MQTT κανάλια (topics) και να παρακολουθούμε όλα τα μνήματα που ανταλλάσσονται μεταξύ του διακομιστή και των συσκευών. Επίσης μπορούμε να δημοσιεύσουμε και εμείς κάποιο μήνυμα μέσω της πλατφόρμας χειροκίνητα σε κάποιο κανάλι. Ο λόγος που χρησιμοποιήθηκε αυτό το πρόγραμμα είναι ότι ήταν ιδιαίτερα χρήσιμο για την δοκιμή των συσκευών αφού είχαν προγραμματιστεί και για την παρακολούθηση των αντιδράσεων του διακομιστή, πριν προχωρήσουμε το σύστημα παρακάτω.

4.4 Υλοποίηση

Σε αυτό το σημείο θα γίνει μία αναλυτική περιγραφή, βήμα προς βήμα, της υλοποίησης της εργασίας για κάθε κομμάτι του συστήματος. Θα αναλύσουμε το στήσιμο του MQTT διακομιστή, την υλοποίηση των κυκλωμάτων ελέγχου των συσκευών, τον προγραμματισμό των μικροελεγκτών, την εφαρμογή μεθόδων, όπως το OTA, στο σύστημα, αλγόριθμους που λύνουν προβλήματα από την μεριά του hardware, την δημιουργία καναλιών επικοινωνίας στον server και γενικότερα όλους τους κανόνες και μεθόδους αυτοματοποίησης.

4.4.1 Εγκατάσταση και Διαμόρφωση MQTT Server

Όπως προαναφέραμε, τον ρόλο του διακομιστή στο σύστημα τον έχει αναλάβει ένας μικροϋπολογιστής Raspberry PI 3 Model B και το λειτουργικό που επιλέξαμε να τρέχει είναι το openHABian, καθώς έχει από δικού του όλα τα κατάλληλα πακέτα του openHAB που θα χρειαστούμε και δε θα χρειαστεί να τα κατεβάσουμε μόνοι μας χειροκίνητα για να τα εγκαταστήσουμε, με κίνδυνο να μας ξεφύγει κάτι ή να προκύψει κάποιο σφάλμα.

Η έκδοση του λειτουργικού που χρησιμοποιήθηκε είναι η 1.3, που κατά την διάρκεια της υλοποίησης ήταν η πιο πρόσφατη, και είναι διαθέσιμη για μεταφόρτωση στο github.com και πιο συγκεκριμένα στον σύνδεσμο https://github.com/openhab/openhabian/releases, όπου μαζί με την 1.3 είναι διαθέσιμες και όλες οι προηγούμενες εκδόσεις αλλά και οι νεότερες που θα προκύψουν στο μέλλον. Μαζί με την έκδοση πάντα είναι διαθέσιμος και ο πηγαίος κώδικας του λειτουργικού.

Το λειτουργικό σύστημα που κατεβάσαμε είναι μορφής image και γι' αυτό παράλληλα μεταφορτώσαμε και το εργαλείο Win32DiskImager για να γράψουμε το λειτουργικό στον server μας. Ο διακομιστής χρησιμοποιεί για αποθηκευτικό χώρο μία κάρτα μνήμης microSD και συνίσταται αυτή να είναι τουλάχιστον 16 GB. Αφού γράψουμε το λειτουργικό σύστημα στην κάρτα μνήμης πρέπει να την βάλουμε στο Raspberry PI 3 και να το εκκινήσουμε. Έπειτα περιμένουμε από 45 λεπτά έως μιάμιση ώρα για να γίνει η επέκταση (expand) του λειτουργικού. Μετά την επέκταση ο διακομιστής μας έχει λειτουργικό και τρέχει κανονικά.

Σε ένα τέτοιο σύστημα οικιακού αυτοματισμού, είναι άκρως σημαντικό και αναγκαίο ο διακομιστής να έχει σταθερή IP. Γι' αυτό πριν προχωρήσουμε παρακάτω πρέπει να πάμε στην σελίδα του δρομολογητή, που συνήθως είναι στην διεύθυνση 192.168.1.1 και από 'κει να βρούμε την φυσική διεύθυνση του Raspberry PI 3 και να της δώσουμε μία static IP που επιθυμούμε. Μετά το IP reservation συνίσταται η επανεκκίνηση του δρομολογητή και η επανεκκίνηση του διακομιστή.

Αφού έχουμε πλέον έναν μικροϋπολογιστή με λειτουργικό σύστημα και κατοχυρωμένη IP μένει να ρυθμίσουμε και να αναβαθμίσουμε το openHABian και να εγκαταστήσουμε τον MQTT server. Πρώτα όμως θα επισκεφτούμε τον μικροϋπολογιστή με έναν browser γράφοντας την IP διεύθυνσή του ή το domain name του (openhabianpi) χρησιμοποιώντας την πόρτα 8080 (openhabianpi:8080). Εκεί μπορούμε να διαλέξουμε τις διεπαφές που προσφέρει το openHAB και να τις εγκαταστήσουμε. Μετά την εγκατάσταση των διεπαφών προσεγγίζουμε τον server με SSH χρησιμοποιώντας το PuTTY, πλέον. Με την χρήση SSH έχουμε πρόσβαση στην γραμμή εντολών του λειτουργικού συστήματος του διακομιστή και με την χρήση του εργαλείου openhabian-config κάνουμε update και upgrade τον διακομιστή και ρυθμίζουμε τις κατάλληλες παραμέτρους όπως time zone, WiFi settings
κ.α.

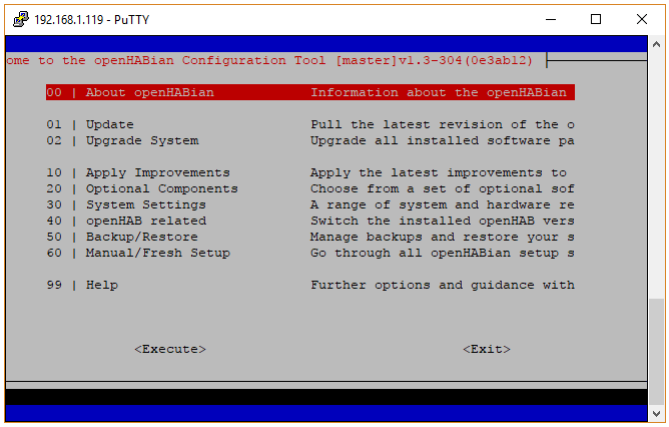

Figure 4-20 openhabian-config

Επίσης σε αυτό το σημείο με την χρήση του ίδιου εργαλείου μπορούμε να στήσουμε αυτόματα τον MQTT server πατώντας στην κατάλληλη επιλογή. Έπειτα εγκαθιστούμε και το αντίστοιχο MQTT binding που βρίσκουμε στο PaperUI. Οι συνιστώσες του MQTT που θα χρειαστούμε στην συνέχεια μπορούμε να τις βρούμε σε ένα αρχείο που θα δημιουργηθεί στο path /etc/openhab2/services/mqtt.cfg. Σε αυτό το αρχείο καταχωρούμε την IP διεύθυνση του διακομιστή και την πόρτα στην οποία θα ακούει το MQTT. Οι πόρτες που χρησιμοποιεί το MQTT είναι οι εξής:

- **1883**: MQTT χωρίς κρυπτογράφηση.
- **8883**: MQTT με κρυπτογράφηση.
- **8884**: MQTT με κρυπτογράφηση, απαιτείται πιστοποιητικό από client.
- **8080**: MQTT μέσω websockets, χωρίς κρυπτογράφηση.
- **8081**: MQTT μέσω websockets, με κρυπτογράφηση.

Η πόρτα στην δική μας περίπτωση είναι η 1883. Άλλες ρυθμίσεις μπορούν να γίνουν στο αρχείο mqtt-eventbus.cfg στο ίδιο directory. Εκεί μπορούμε να ρυθμίσουμε το όνομα του broker και άλλες παραμέτρους για τα κανάλια (topics) επικοινωνίας. Στην δικιά μας περίπτωση αφήνουμε αυτές τις ρυθμίσεις στην αρχική τους κατάσταση, καθώς δεν είναι τόσο χρήσιμες ή αναγκαίες για την παρούσα εργασία.

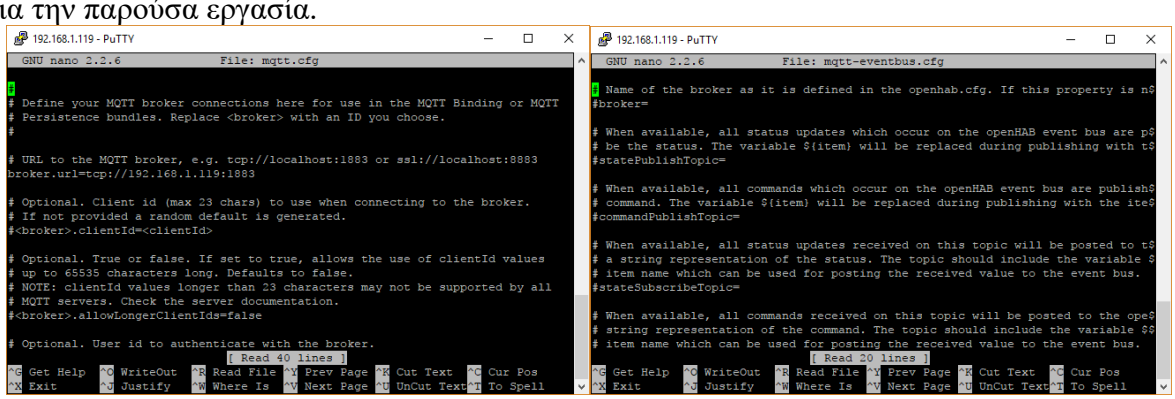

Figure 4-21 mqtt.cfg & mqtt-eventbus.cfg

Καλό είναι να αλλάξουμε και τον αρχικό κωδικό για login στο σύστημα. Αυτό γίνεται με την χρήση του passwd στην γραμμή εντολών. Για την αλλαγή του samba password χρησιμοποιούμαι την εντολή sudo smbpasswd openhabian.

Τέλος μένει να εγκαταστήσουμε από το PaperUI το cloud connector. Οπότε πάμε στα Add-ons στην διεπαφή και επιλέγουμε το κατάλληλο binding και εκτελούμε την διαδικασία που περιγράφεται στο κεφάλαιο 3. Για την σύνδεση του διακομιστή με το cloud. Πλέον, ο MQTT διακομιστής είναι έτοιμος και μένει μόνο να δημιουργηθούν οι διεπαφές του χρήστη και να εγκαθιδρυθεί η επικοινωνία μεταξύ server και συσκευών. Αυτό πιο αναλυτικά γίνεται στο κεφάλαιο 4.3.4.

4.4.2 Ηλεκτρονικά Κυκλώματα

Σε αυτό το σημείο αφήνουμε την διαμόρφωση του διακομιστή προς το παρόν και ξεκινάμε την δημιουργία των κυκλωμάτων για τον έλεγχο των συσκευών και τις κατάλληλες συνδέσεις για τον προγραμματισμό των μικροελεγκτών και έπειτα την συνδεσμολογία για την εκτέλεση του firmware που τους έχουμε φορτώσει.

Σύστημα Ασφαλείας

Το κύκλωμα για την ασφάλεια του σπιτιού είναι ένα βασικό κύκλωμα με την χρήση ενός magnetic reed switch που ενσωματώνεται πάνω στις πόρτες που μας ενδιαφέρουν. Παρ' όλα αυτά με την κατάλληλη χρήση κανόνων αυτοματοποίησης στη μηχανή κανόνων του openHAB στον διακομιστή μας μπορεί να γίνει ένα αρκετά έξυπνο και ευέλικτο σύστημα ασφάλειας που μπορεί να αποκτήσει και επιπλέον λειτουργίες για έλεγχο των συσκευών μέσα στο σπίτι.

Το κύκλωμα για λόγους ευκολίας πραγματοποιήθηκε πάνω σε ένα breadboard και με την χρήση του μικροελεγχτή ESP8266 – 12F. Για πραγματική χρήση το κύκλωμα αυτό τυπώνεται σε ένα PCB και ενσωματώνεται στην πόρτα και ένας μαγνήτης στην κάσα της πόρτας για την λειτουργεία του μαγνητικού διακόπτη Επιπλέον χρησιμοποιήθηκες ένα USB – to – Serial converter για τον προγραμματισμό του από τον ηλεκτρονικό υπολογιστή. Το κύκλωμα είναι το εξής:

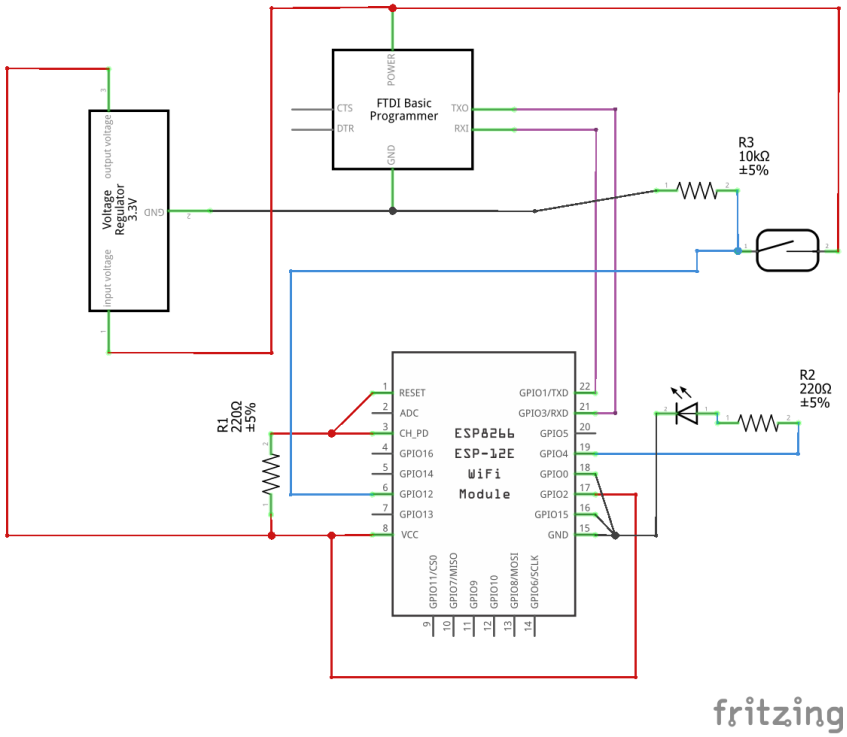

Figure 4-22 Κύκλωμα συστήματος ασφαλείας

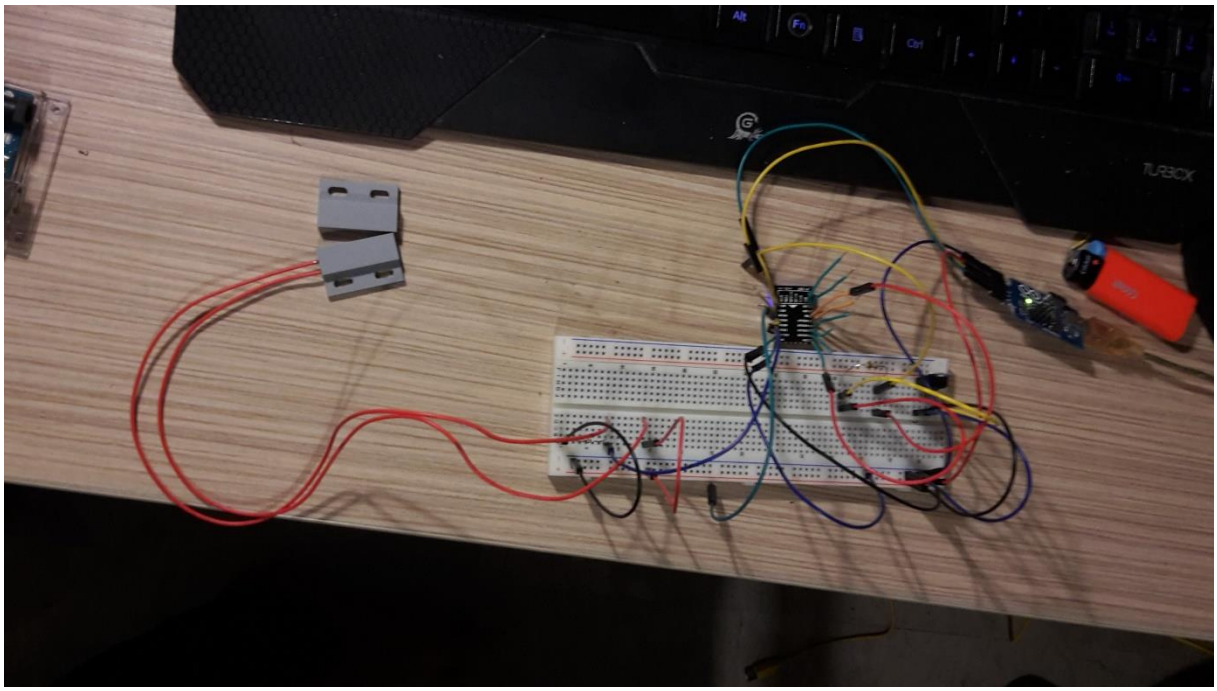

Figure 4-23 Υλοποιημένο κύκλωμα ασφαλείας

Ο μικροελεγκτής δουλεύει στα 3.3 Volts σύμφωνα με τον κατασκευαστή, οπότε είναι αναγκαίο να χρησιμοποιήσουμε ένα voltage regulator για την προστασία του από τα 5 Volt που δίνει το USB – to – Serial converter.

Η συνδεσμολογία είναι η ακόλουθη: στα pins RST (reset), CH_EN (enable chip), VCC και GPIO2 δίνουμε τάση. Τα pins GND (ground), GPIO15 τα γειώνουμε. Το pin GPIO0 πρέπει να είναι γειωμένο όταν θέλουμε να προγραμματίσουμε το chip (UART mode) και not connected ή HIGH όταν θέλουμε να είναι σε FLASH BOOT mode, δηλαδή, να τρέχει το πρόγραμμα που του έχουμε φορτώσει. Τον μαγνητικό διακόπτη τον έχουμε συνδέσει στην

GPIO12 και του δίνουμε κανονικά τάση 5 Volt μέσω μίας αντίστασης 10 ΚΩ. Στην GPIO4 έχουμε συνδεδεμένο ένα LED το οποίο αναβοσβήνει όταν έχουμε ενεργοποιήσει στον διακομιστή το σύστημα ασφαλείας. Το σύστημα ασφαλείας δεν είναι πάντα ενεργό καθώς δεν έχει νόημα όταν βρισκόμαστε εντός του σπιτιού. Μέσω των θυρών Tx και Rx επικοινωνεί ο μικροελεγκτής με τον USB – to – Serial converter για την πρώτη φόρτωση του firmware. Η σύνδεση είναι Tx -> Rx, Rx -> Tx μεταξύ του USB και του μικροελεγχτή. Αφού φορτώσουμε το πρόγραμμα για πρώτη φορά τα καλώδια αυτά περισσεύουν καθώς οι επόμενες ενημερώσεις θα γίνουν με την χρήση OTA. Η τάση και η γείωση που μας παρέχει το converter μπορεί να αντικατασταθεί με μία μπαταρία μετά την εγκατάσταση του κυκλώματος.

Σύστημα Ελέγχου με Ρελέ

Το συγκεκριμένο σύστημα δεν είναι ιδιαίτερα δύσκολο στην υλοποίησή του όσον αφορά την συνδεσμολογία καθώς είναι ένα σύστημα δύο καταστάσεων και χρησιμοποιήθηκε η πλακέτα ESP8266 – EVB της Olimex με ενσωματωμένο ρελέ. Αυτό το σύστημα υλοποιήθηκε για τον έλεγχο του φωτισμού, αλλά με πολύ μικρές αλλαγές σε ορισμένα σημεία του κώδικα έως καθόλου μπορεί να χρησιμοποιηθεί σε οποιαδήποτε συσκευή δύο καταστάσεων μέσα σε ένα σπίτι όπως για παράδειγμα για τον έλεγχο του θερμοσίφωνα ή του ποτιστικού συστήματος. Λαμβάνοντας υπ’ όψη και τον προγραμματισμό της πλακέτας έχουμε τεράστια ευελιξία στην διαχείριση οικιακού εξοπλισμού δύο καταστάσεων καθώς μπορούμε να προσθέσουμε χρονοδιαγράμματα και κανόνες αυτοματοποίησης στην μηχανή κανόνων του openHAB.

Ένας εξοπλισμός δύο καταστάσεων όμως χρειάζεται και διακόπτες φυσικούς πέρα των εικονικών που δημιουργούμε στις διεπαφές. Γι’ αυτόν τον λόγο πραγματοποιήθηκε ένα κύκλωμα με ένα ψηφιακό διακόπτη με την χρήση ενός pushbutton το οποίο επικοινωνεί με την GPIO16 του μικροελεγχτή μέσω των pins της πλακέτας. Στις παρακάτω φωτογραφίες βλέπουμε την συνδεσμολογία που έχει γίνει.

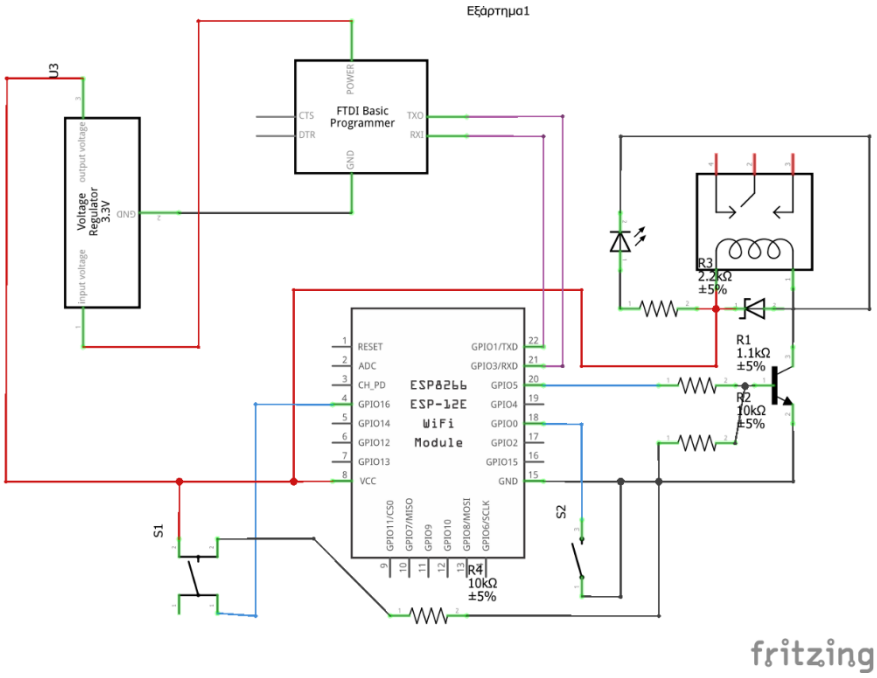

Figure 4-24 Κύκλωμα συστήματος με ρελέ

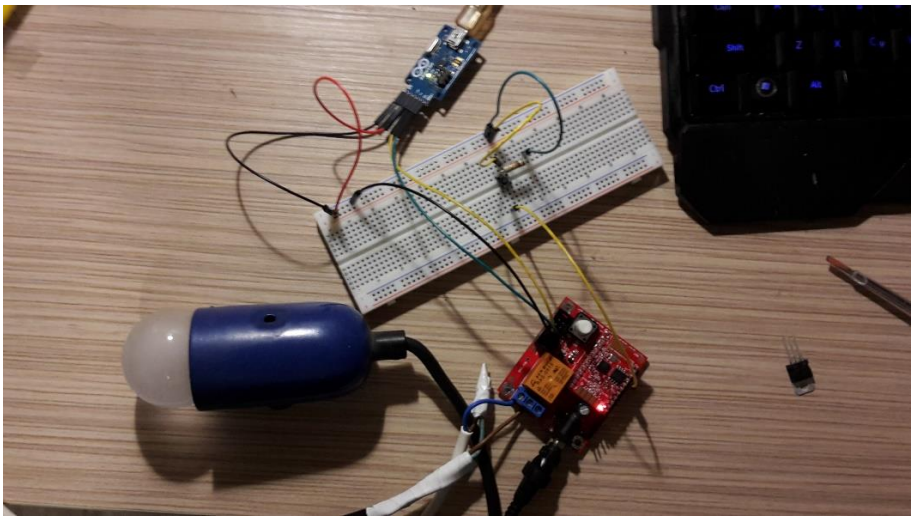

Figure 4-25 Υλοποιημένο κύκλωμα συστήματος ρελέ

Το UEXT της πλακέτας έχει συνδεθεί με το USB – to – Serial converter όπως και στον μικροελεγχτή ESP8266 – 12F για την φόρτωση του firmware στον μικροελεγχτή. Μετά τον προγραμματισμό της συσκευής μπορούμε να την τροφοδοτούμε με συνεχές ρεύμα τάσης 5 Volt από την mini – jack υποδοχή ή με την σύνδεση κάποιας μπαταρίας στην GPIO1 και GPIO2 αφού βέβαια ρίξουμε την τάση στα 3.3 Volt με την χρήση ενός ρυθμιστή τάσης καθώς το pin της πλακέτας επικοινωνεί άμεσα με το ESP8266 και δεν ρυθμίζεται η τάση όπως γίνεται με την υποδοχή συνεχούς ρεύματος.

Με την χρήση του pushbutton στην GPIO16 μπορούμε να ελέγχουμε τον εξοπλισμό «χειροκίνητα» και με την χρήση του ρελέ της πλακέτας που επικοινωνεί με την GPIO5 του μικροελεγκτή μπορούμε να διαχειριστούμε τον εξοπλισμό μέσω της διεπαφής.

Ένα μειονέκτημα της συγκεκριμένης πλακέτας είναι πως έχει 2 MB Flash Memory με αποτέλεσμα να μην συνίσταται η χρήση OTA. Αυτό μπορεί να μας ταλαιπωρήσει σε περίπτωση που επιθυμούμε να επαναπρογραμματίσουμε την συσκευή γιατί θα πρέπει να έχουμε φυσική πρόσβαση στην συσκευή, να την αφαιρέσουμε από τον εξοπλισμό, να την συνδέσουμε στον υπολογιστή, να την επαναπρογραμματίσουμε και έπειτα να την επανασυνδέσουμε στον εξοπλισμό και τέλος στην περίπτωση που δεν δουλεύει καλά το firmware πρέπει να κάνουμε την ίδια διαδικασία ξανά. Γι’ αυτό καλό είναι όταν χρησιμοποιούμαι αυτή την συσκευή να είμαστε σίγουροι για το firmware και να βρίσκεται σε κάποιο σημείο με εύκολη φυσική πρόσβαση για την περίπτωση μελλοντικής ενημέρωσης.

Κύκλωμα Ελέγχου Φωτιστικών με Dimmer

Για τον έλεγχο των φωτιστικών συσκευών ενός έξυπνου σπιτιού δημιουργήθηκαν δύο κυκλώματα, ένα συνεχούς και ένα εναλλασσόμενου ρεύματος, που επικοινωνούν μεταξύ τους μέσω των λεγόμενων opto – isolators (ή opto – coupler). Το κύκλωμα φαίνεται στην παρακάτω φωτογραφία.

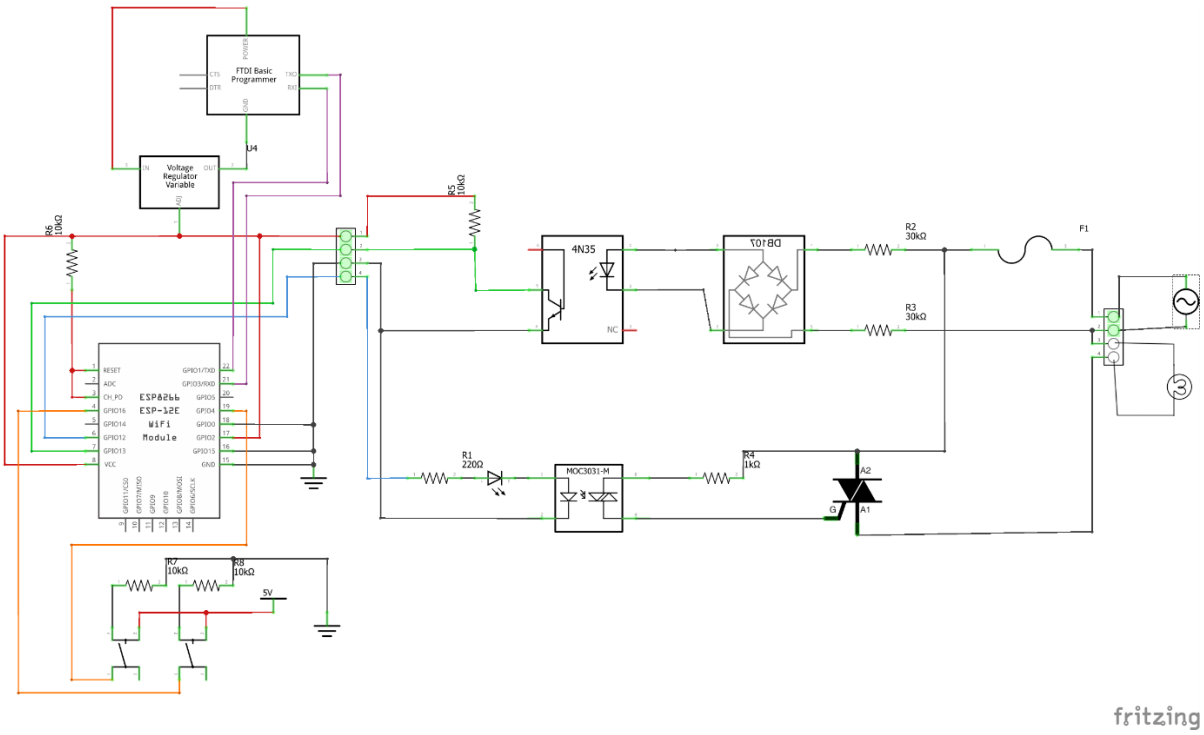

Figure 4-26 Κύκλωμα dimmer

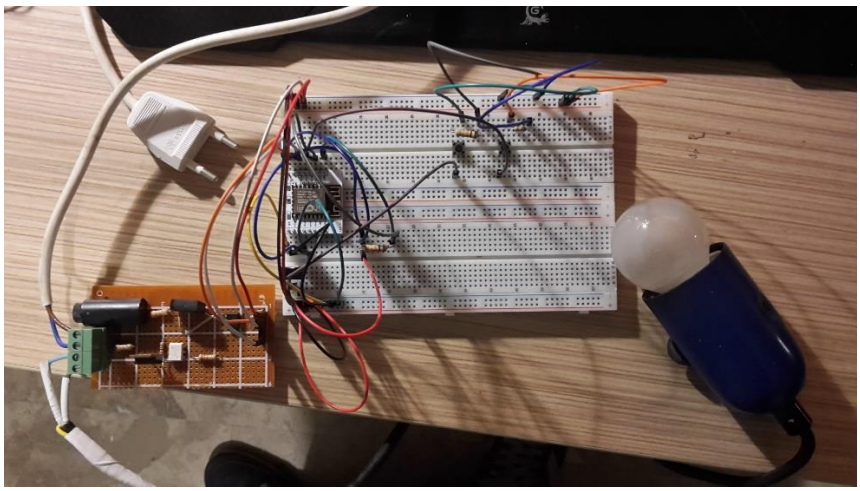

Figure 4-27 Υλοποιημένο κύκλωμα dimmer

Για να μπορέσουμε να κάνουμε dimming σε εναλλασσόμενο ρεύμα υπάρχουν τρεις τρόποι. Phase Cutting, Pulse Skip Modulation και Pulse Width Modulation (PWM) dimming. Πιο αναλυτικά:

- **Phase Cutting:**

  Ένας τρόπος για να το πετύχουμε είναι μέσω του ελέγχου φάσης με ένα TRIAC: Το TRIAC, τότε είναι ανοιχτό, αλλά μόνο για ένα μέρος του κύματος του εναλλασσόμενου ρεύματος. Αυτό λέγεται leading edge cutting.

  Θα μπορούσαμε να αφήσουμε τον μικροελεγκτή να ανοίγει το TRIAC για λίγα microseconds, αλλά αυτό έχει το πρόβλημα ότι είναι μη προβλέψιμο σε ποιο μέρος του ημιτόνου ανοίγει το TRIAC με αποτέλεσμα το επίπεδο του dimming να είναι απρόβλεπτο. Αυτό σημαίνει ότι χρειαζόμαστε κάποιο σημείο αναφοράς.

  Για αυτό τον λόγο ένας ανιχνευτής zero crossing είναι απαραίτητος. Αυτός ειδοποιεί τον μικροελεγκτή πότε στο κύμα του ημιτόνου περνάει από το μηδέν και μας δίνει ένα καθορισμένο σημείο πάνω στην ημιτονοειδή συνάρτηση.

- **Pulse Skip Modulation (PSM):**

  Ένας άλλος τρόπος για να το κάνουμε είναι το PSM . Με το PSM, ένας ή περισσότεροι ολοκληρωμένοι κύκλοι (ημίτονα) μεταφέρονται στο φορτίο και έπειτα ένα ή περισσότεροι κύκλοι παραλείπονται. Αν και αποτελεσματικό, δεν είναι ένας καλός τρόπος για dimming καθώς υπάρχει η πιθανότητα η λάμπα να αναβοσβήνει. Στο PSM πρέπει πάντα να αφήνουμε όλο τον κύκλο της συνάρτησης να περνάει και όχι τον μισό καθώς σε αυτή την περίπτωση το φορτίο θα τροφοδοτείται πραγματικά από DC, πράγμα το οποίο δεν είναι καλό για AC φορτία. Η διαφορά μεταξύ του Phase cutting και PSM είναι κυρίως στο λογισμικό, καθώς και στις δύο περιπτώσεις θα χρειαστούμε ένα κύκλωμα που εντοπίζει zero – crossing και μπορεί να ελέγξει ένα TRIAC.

  Ένα κύκλωμα που μπορεί να το κάνει αυτό είναι εύκολο να φτιαχτεί. Το zero crossing προέρχεται άμεσα από το ανορθωμένο εναλλασσόμενο ρεύμα – μέσω ενός opto – coupler φυσικά – και δίνει ένα σήμα κάθε φορά που το ημίτονο περνάει από το 0. Επειδή το ημίτονο περνάει πρώτα μέσα από ανόρθωση διπλής φάσης, το zero crossing σήμα δίνεται ανεξάρτητα από το αν το ημίτονο αυξάνεται προς το 0 ή μειώνεται προς το 0. Αυτό το σήμα μπορεί να χρησιμοποιηθεί σε ένα interrupt (διακοπή) στον μικροελεγκτή.

- **PWM Dimming:**

  Το PWM dimming, σε αντίθεση με τα LEDs, δε χρησιμοποιείται συχνά με φορτία εναλλασσόμενου ρεύματος για διάφορους λόγους. Παρ' όλα αυτά είναι δυνατόν. Για να το επιτύχουμε χρειαζόμαστε γαλβανικό διαχωρισμό μεταξύ του κυκλώματος του μικροελεγχτή και του κυκλώματος που είναι συνδεδεμένο με το εναλλασσόμενο ρεύμα, γι' αυτό και χρειαζόμαστε τους οπτοσυζευκτές.

  Το εικονιζόμενο κύκλωμα κάνει ακριβώς αυτό. Το ρεύμα των 220V οδηγείται μέσω δύο αντιστάσεων 30kΩ σε μία γέφυρα ανορθωτή που δίνει ένα ανορθωμένο σήμα διπλής φάσης σε ένα 4ν35 optocoupler. Το LED στο optocoupler γίνεται low με συχνότητα 100MHz και το σήμα στον συλλέκτη γίνεται high με συχνότητα 100MHz, σύμφωνα με το ημιτονοειδές κύμα από το ηλεκτρικό δίκτυο. Το σήμα από το 4n35 τροφοδοτείται σε ένα interrupt pin στον μικροελεγκτή. Η συνάρτηση interrupt τροφοδοτεί με ένα σήμα συγκεκριμένου μήκους κάποιο από τα GPIOs. Το σήμα αυτό επιστρέφει στο AC κύκλωμα και ανοίγει το MOC3021 που ενεργοποιεί το opto – thyristor για ένα σύντομο χρονικό διάστημα.

Όλη αυτή η διαδικασία μπορεί να γίνει με την χρήση μίας συμβατικής λάμπας καθώς αυτές είναι αρκετά κατάλληλες για τέτοια χρήση. Μπορεί να γίνει και με λάμπα αλογόνου αλλά θα μειώσει τον χρόνο ζωής της. Για cfl και LED λάμπες δεν θα δουλέψει εκτός και αν είναι φτιαγμένες ειδικά για τέτοια χρήση.

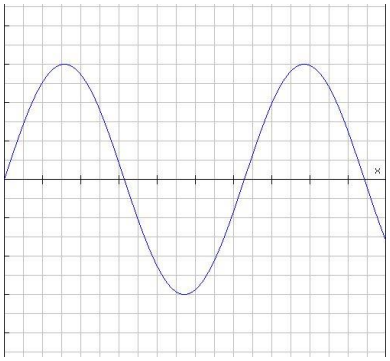

Figure 4-31 AC Input

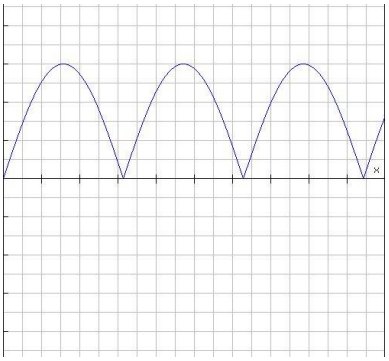

Figure 4-30Fully Rectified AC

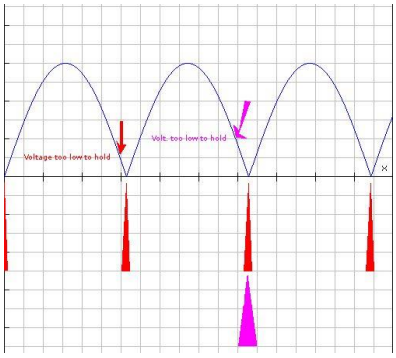

Figure 4-29 Zero Crossing

4.4.3 Προγραμματισμός Μικροελεγκτών

Για να προγραμματίσουμε μικροελεγχτές με chip ESP8266EX στο περιβάλλον Arduino IDE πρέπει πρώτα να εφαρμόσουμε κάποιες διαδικασίες. Για αρχή εγκαθιστούμε στον διαχειριστή πλακετών το πακέτο esp8266 από ESP8266 Community που περιλαμβάνει όλες τις πλακέτες που θα χρειαστούμε για αυτή την εργασία. Έπειτα να σιγουρευτούμε πως είναι εγκατεστημένες όλες οι βιβλιοθήκες που αφορούν το chip ESP8266EX καθώς και οι βιβλιοθήκες ArduinoOTA, MQTT, OneButton οι οποίες χρειάζονται για τις υπόλοιπες πτυχές των firmware που θα γράψουμε. Την ArduinoOTA για την εφαρμογή του OTA, την MQTT για τα MQTT μηνύματα μεταξύ server και clients και την βιβλιοθήκη OneButton για τις συναρτήσεις των πραγματικών διακοπτών στο πρόγραμμα για το dimmer. Τέλος, στις προτιμήσεις πρέπει να προσθέσουμε το URL http://arduino.esp8266.com/stable/package_esp8266com_index.json στην θέση «Επιπλέον URLs διαχειριστή πλακετών».

Προγραμματισμός Συστήματος με Ρελέ

Το σύστημα ελέγχου εξοπλισμού με δύο καταστάσεις υλοποιήθηκε με την χρήση της πλακέτας ESP8266 – EVB και δεν περιέχει την δυνατότητα ενμέρωσης με OTA καθώς η μνήμη flash του chip είναι στα 2MB και το OTA συστήνεται για flash μνήμη 4MB, όπως διαθέτουν τα ESP8266 – 12E/F.

Στην αρχή του προγράμματος δηλώνουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε σε αυτό το πρόγραμμα και είναι οι εξής:

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

Η βιβλιοθήκη esp8266WiFi.h ενεργοποιεί την δυνατότητα σύνδεσης στο δίκτυο (Internet και τοπικά) χρησιμοποιώντας το ενσωματωμένο WiFi του chip και η MQTTClient.h χρησιμοποιείται για την ανταλλαγή των MQTT μνημάτων. Έπειτα καθορίζουμε της ρυθμίσεις του WiFi και του MQTT. Δηλαδή φτιάχνουμε σταθερές που περιέχουν το ssid και το password του WiFi που επιθυμούμε να συνδεθούμε και ορίζουμε την IP διεύθυνση του server που θα επικοινωνεί η συσκευή μας και τα topics μέσω των οποίων θα γίνονται οι ανταλλαγές μνημάτων και ένα μοναδικό αναγνωριστικό για την συσκευή μας.

```
/*-----------------------SETTINGS-----------------------*/

/*WiFi Settings*/

const char* ssid = "ssid";
const char* password = "password";

/*MQTT Settings*/

char* lightsTopic = "MyHome/lights/MBR_Light";
char* rssiTopic = "MyHome/rssiLights/MBR_Light";
const char* server = "192.168.1.119";
const char* mqttDeviceID = "Device1";
/*------------------------------------------------------
```

Τέλος δηλώνουμε τις μεταβλητές που αφορούν το ρελέ, το οποίο βρίσκεται στην GPIO 5, και το pushbutton που το έχουμε συνδέσει με την GPIO16, μαζί με τις μεταβλητές που είναι

απαραίτητες για την διαχείρισή του. Επιπλέον όλες τις μεταβλητές που θα χρειαστούμε για την αποστολή MQTT μηνυμάτων και την εγκαθίδρυση του MQTT client και της συνάρτησης που συνδέει τον μικροελεγκτή μας στο δίκτυο.

```
//Relay
int relay = 5;
int relayState = LOW;

//Button
const int button = 16;
int buttonState = LOW;
int lastButtonState = HIGH;
int reading;
long lastDebounceTime = 0;
long debounceDelay = 50;

//MQTT messages for server
char lightMsg[32];
char rssiMsg[32];

//MQTT
WiFiClient net;
MQTTClient client;
//Function to connect to WiFi
void connect();
```

Επόμενο βήμα είναι να φτιάξουμε την μία εκ των δύο βασικών συναρτήσεων του Arduino IDE, την void setup(). Σε αυτή την συνάρτηση θα δηλώσουμε το mode των pin του μικροελεγχτή, δηλαδή αν θα χρησιμοποιούνται για INPUT ή OUTPUT και θα αρχικοποιήσουμε την κατάσταση του pin που συνδέεται το ρελέ σε LOW, δηλαδή σαν αρχική κατάσταση να μην ανάβει το φως. Επιπλέον Ενεργοποιούμε την σύνδεση στο WiFi και την εκκίνηση της λειτουργείας του μικροελεγκτή σαν MQTT client και του δίνουμε την συνάρτηση που θα εκτελεί στην λήψη μηνυμάτων. Τέλος, κάνουμε κλήση της συνάρτησης void connect() για την εγγραφή στο topic από το οποίο θα λαμβάνονται τα μηνύματα και θα ελέγχουμε την κατάσταση σύνδεσης στο WiFi και την σύνδεση της συσκευής στο κανάλι.

```
void setup() {
  pinMode(relay, OUTPUT);
  digitalWrite(relay, LOW);
  pinMode(button, INPUT);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.begin(server, net);
  client.onMessage(messageReceived);

  connect();
}

void connect() {
  while(WiFi.status() != WL_CONNECTED) {
```

```
    delay(1000);
  }

  while(!client.connect(mqttDeviceID)) {
    delay(1000);
  }

  client.subscribe(lightsTopic);
}
```

Μετά την void setup() και void connect(), γράφουμε την δεύτερη από τις δύο βασικότερες συναρτήσεις του Arduino IDE την void loop(). Σε αυτή τη συνάρτηση πραγματοποιείται όλη η λειτουργεία που επιθυμούμε για τον εξοπλισμό με τον οποίο είναι συνδεδεμένος ο μικροελεγχτής. Για αρχή, η συνάρτηση client.loop(), αφού συνδεθεί ο μικροελεγκτής, επιτρέπει στον client να διατηρεί την σύνδεσή του και να ελέγχει τακτικά για εισερχόμενα MQTT μηνύματα. Έπειτα γίνεται τακτικά έλεγχος για την κατάσταση σύνδεσης της συσκευής στο δίκτυο και γράφονται οι κατάλληλες εντολές για το πως πρέπει να ανταποκριθεί ο μικροελεγκτής με την χρήση του pushbutton. Καθώς το pushbutton το χρησιμοποιούμε σαν διακόπτη πρέπει κάθε φορά που αλλάζει η κατάστασή του για ελάχιστο χρονικό διάστημα από την δικιά μας, ανθρώπινη, αντίληψη, και μετά επανέρχεται στην προηγούμενή κατάσταση, μόνο τότε να συμβαίνει η επιθυμητή αλλαγή. Από άποψη hardware όμως αυτό δημιουργεί το λεγόμενο πρόβλημα αναπήδησης.

Αναλυτικά, η αναπήδηση (bouncing) είναι η τάση δύο οποιονδήποτε μεταλλικών επαφών σε μία ηλεκτρονική συσκευή να δημιουργούν πολλαπλά σήματα καθώς οι επαφές πλησιάζουν ή απομακρύνονται. Το debouncing είναι οποιαδήποτε μέθοδος χρησιμοποιείται από άποψη hardware ή software που μας εξασφαλίζει ότι θα λαμβάνεται δράση για ένα μόνο σήμα από αυτά που δημιουργούνται για κάθε άνοιγμα ή κλείσιμο της επαφής.

Για παράδειγμα όταν πατάμε κάποιο πλήκτρο στο πληκτρολόγιο του ηλεκτρονικού υπολογιστή, περιμένουμε μία μόνο επαφή να καταγραφεί από τον υπολογιστή. Στην πραγματικότητα όμως υπάρχει μία αρχική επαφή, .ία ελαφριά αναπήδηση της επαφής, μετά άλλη μία επαφή καθώς η αναπήδηση τελειώνει και μετά άλλη μία αναπήδηση πίσω κ.ο.κ. ένα παρόμοιο φαινόμενο λαμβάνει χώρα όταν ανοίγει ένας διακόπτης που αποτελείται από μεταλλικές επαφές.

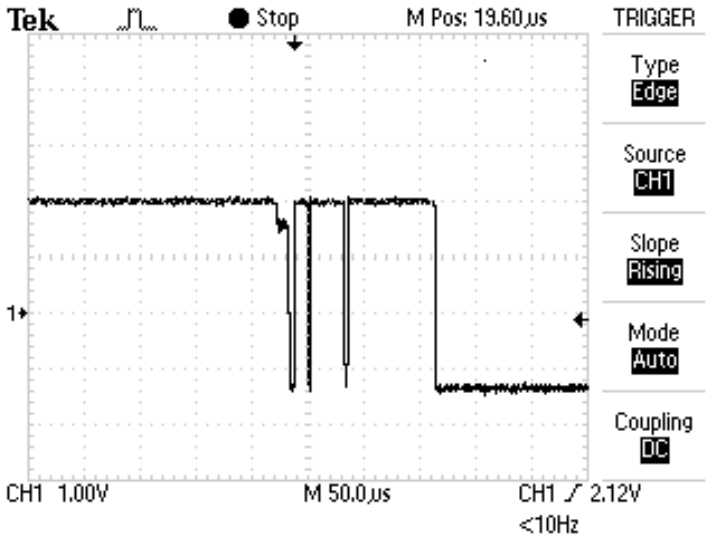

Figure 4-32 Switch Bounce

Η συνήθης λύση είναι ένα hardware ή ένα λογισμικό αποκατάστασης που εξασφαλίζει ότι μπορεί να καταχωρηθεί μόνο ένα ψηφιακό σήμα μέσα σε ένα συγκεκριμένο χρόνο (συνήθως χιλιοστά του δευτερολέπτου).

Στην εργασία αυτήν χρησιμοποιούνται pushbuttons στα κυκλώματα ελέγχου των ηλεκτρικών λαμπτήρων για πραγματικούς διακόπτες· με αποτέλεσμα να δημιουργούνται πολλαπλά σήματα από την αναπήδηση και να ξεγελάνε τον μικροελεγχτή και επομένως να μην έχουμε τα επιθυμητά αποτελέσματα. Η λύση που χρησιμοποιούμε για τα pushbuttons είναι η εξής:

```
reading = digitalRead(button);
if(reading != lastButtonState){
  lastDebounceTime = millis();
  lastButtonState = reading;
}

if((millis() - lastDebounceTime) > debounceDelay){
  if(buttonState != lastButtonState){
    buttonState = lastButtonState;
    if(buttonState == HIGH){
      relayState = !relayState;
      digitalWrite(relay, relayState);
      sprintf(lightMsg, "%d", relayState);
      client.publish(lightsTopic, lightMsg);
    }
  }
}
```

Το πρόγραμμα αυτό πιο απλά, μετράει σε περίπτωση αλλαγής της κατάστασης του pushbutton την διάρκεια που κράτησε αυτή η αλλαγή. Αν η διάρκεια αυτή είναι μεγαλύτερη από 50 χιλιοστά του δευτερολέπτου και η αλλαγή αυτής της κατάστασης ήταν από LOW σε HIGH, τότε και μόνο το σήμα θεωρείται αυθεντικό και αλλάζει η κατάσταση του pin που είναι συνδεδεμένο το ρελέ. Όλη η συνάρτηση void loop() έχει ως εξής:

```
void loop() {
  client.loop();
  delay(10);

  if(!client.connected())
    connect();

  /*--------------BUTTON-SWITCH---------------*/
  reading = digitalRead(button);
  if(reading != lastButtonState){
    lastDebounceTime = millis();
    lastButtonState = reading;
  }

  if((millis() - lastDebounceTime) > debounceDelay){
    if(buttonState != lastButtonState){
      buttonState = lastButtonState;
      if(buttonState == HIGH){
```

```
        relayState = !relayState;
        digitalWrite(relay, relayState);
        sprintf(lightMsg, "%d", relayState);
        client.publish(lightsTopic, lightMsg);
      }
    }
  }
  /*----------------------------------------*/
}
```

Στο τέλος του προγράμματος έχει μείνει η void messageReceived(“topic”, “message”) συνάρτηση η οποία καλείται όταν η client.loop() εντοπίζει MQTT μήνυμα στο topic που είναι γραμμένος ο μικροελεγχτής και μέσα σε αυτή γράφονται οι κατάλληλες εντολές για την υλοποίηση των ενεργειών που επιθυμούμε από τον μικροελεγχτή για την διαχείριση του εξοπλισμού. Στο δικό μας παράδειγμα δηλαδή, την αλλαγή της κατάστασης του ρελέ. Πιο αναλυτικά στην λήψη του μηνύματος L1ON η κατάσταση του ρελέ γυρνάει σε HIGH, αν είναι LOW και στην λήψη του μηνύματος L1OFF γυρνάει σε LOW, αν είναι HIGH.

```
void messageReceived(String &topic, String &payload) {
  String msg = payload;

  if (msg == "L1ON") {
    digitalWrite(relay, HIGH);
    relayState = !relayState;
    delay(250);
  }

  else if(msg == "L1OFF") {
    digitalWrite(relay, LOW);
    relayState = !relayState;
    delay(250);
  }
}
```

Μαζί με τα άλλα έχουμε προγραμματίσει την συσκευή μας να αποθηκεύει σε μία μεταβλητή το received signal strength indication (RSSI) και να το μετατρέπει dB σε τιμές από 0 – 4 σε μία συνάρτηση που καλούμε, την int db_to_quality(), και η επιστρεφόμενη τιμή δημοσιεύεται στο αντίστοιχο topic, στην περίπτωσή μας το "MyHome/rssiLights/MBR_Light", αφού βέβαια μετατραπεί σε συμβολοσειρά. Η δημοσίευση του RSSI γίνεται κάθε δευτερόλεπτο. Ο κώδικας της συνάρτησης και της δημοσίευσης βρίσκεται μέσα στην void loop() και είναι ο εξής:

```
/*-------------------RSSI-------------------*/
  int rssi = WiFi.RSSI();
  int quality = db_to_quality(rssi);
  //from int to char
  sprintf(rssiMsg, "%d", quality);

  if (millis() - lastMillis > 1000) {
    lastMillis = millis();
    client.publish(rssiTopic, rssiMsg);
  }
```

```
  /*----------------------------------------*/
int db_to_quality(int frssi){
  int q;
 if (frssi >= -50
    return 4;
  else if (frssi <= -100)
    return 0;
  else if (frssi <= -50 && frssi >= -66)
    return 3;
  else if (frssi <= -67 && frssi >= -82)
    return 2;
  else if (frssi <= -83 && frssi >= -99)
    return 1;
}
```

Στο παράρτημα της εργασίας βρίσκεται ο κώδικας ολόκληρος γραμμένος με δομή και σχόλια.

Προγραμματισμός Συστήματος Ασφαλείας

Για τον προγραμματισμό του μικροελεγκτή ESP8266 – 12F για το σύστημα ασφαλείας, οι βιβλιοθήκες όπως και οι ρυθμίσεις που θα χρησιμοποιήσουμε είναι ίδιες με την εφαρμογή του προηγούμενου συστήματος οπότε τα κοινά σημεία θα τα προσπεράσουμε και θα επικεντρωθούμε στα σημεία που διαφέρουν. Ο κώδικας μπορεί να βρεθεί ολόκληρος στο παράρτημα.

Χρησιμοποιούμε τρεις επιπλέον βιβλιοθήκες για να είναι εφικτό το OTA. Αυτές είναι οι ESP8266mDNS.h, WiFiUdp.h και ArduinoOTA.h που αναφέραμε στο προηγούμενο υποκεφάλαιο.

```
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
```

Τα topic που ορίζουμε για την λήψη μηνυμάτων από τον server και για την αποστολή είναι τα εξής:

```
char* sensorTopic = "MyHome/doors/Main_Door";
char* securityTopic = "MyHome/security/Main_Door";
```

Στην δήλωση μεταβλητών δηλώνουμε τον pin που ανταποκρίνεται ο μαγνητικός διακόπτης και θέτουμε επιπλέον τις Boolean μεταβλητές opened, closed και secFlag, καθώς και το LED που σχετίζεται με το αν το σύστημα ασφαλείας είναι ενεργοποιημένο ή όχι. Επίσης επειδή θέλουμε η gpio0 να είναι HIGH πάντα μετά τον πρώτο προγραμματισμό της πλακέτας την δηλώνουμε και αυτή. Απλά κατά την διάρκεια του πρώτου προγραμματισμού με USB πρέπει να είμαστε σίγουροι ότι θα μπει στο UART mode και αυτό το πετυχαίνουμε οδηγώντας με ένα καλώδιο την gpio0 στην γείωση της τροφοδοσίας του κυκλώματος.

```
//Magnetic Reed Switch
int sensor = 12;
boolean opened = true;
boolean closed;

//LED
```

```
int led = 4;
boolean secFlag = false;
int gpio0_pin = 0;
int gpio2_pin = 2;
```

Στην void setup() ορίζουμε την λειτουργεία των pins και τις συναρτήσεις του OTA.

```
// preparing GPIOs
pinMode(gpio0_pin, OUTPUT);
digitalWrite(gpio0_pin, HIGH);
pinMode(gpio2_pin, OUTPUT);
digitalWrite(gpio2_pin, HIGH);

pinMode(sensor, INPUT);
digitalWrite(sensor, LOW);
pinMode(led, OUTPUT);
digitalWrite(led, LOW);
ArduinoOTA.setPassword((const char *)"@dami$0805");

connect();

ArduinoOTA.onStart([]() {
  digitalWrite(led, HIGH);
});
ArduinoOTA.onEnd([]() {
  digitalWrite(led, LOW);
});
ArduinoOTA.onError([](ota_error_t error) {
  ESP.restart();
});
ArduinoOTA.begin();
```

Έπειτα στην void loop() γράφουμε τον κώδικα για να εκτελέσει ο μικροελεγκτής την λειτουργεία που επιθυμούμε, καθώς και την λειτουργεία του OTA. Σε αυτό το πρότζεκτ είναι προγραμματισμένοι οι μικροελεγκτές όταν ανοίγουν να περιμένουν για 15 δευτερόλεπτα σε περίπτωση που έρθει ενημέρωση από το Arduino IDE για να την εγκαταστήσουν. Με το πέρας των 15 δευτερολέπτων συνεχίζει κανονικά το πρόγραμμα μέσα στην void loop() χωρίς να ξαναπηγαίνει στην εντολή ArduinoOTA.handle().

```
void loop() {
  if (ota_flag) {
    while (time_elapsed < 15000) {
      ArduinoOTA.handle();
      time_elapsed = millis();
      delay(10);
    }
    ota_flag = false;
  }
  client.loop();

  if(secFlag){
    digitalWrite(led, HIGH);
```

```
    delay(200);
    digitalWrite(led, LOW);
    delay(200);
  }
  if (!client.connected())
    connect();

  if (digitalRead(sensor) == HIGH) {
    if (opened) {
      opened = false;
      client.publish(sensorTopic, "OPEN");
      delay(50);
    }
    closed = true;
  }
  if (digitalRead(sensor) == LOW) {
    if (closed) {
      closed = false;
      client.publish(sensorTopic, "CLOSED");
      delay(50);
    }
    opened = true;
  }
```

Μέσα στην void loop() το πρόγραμμά μας μετά τις ρουτίνες του WiFi και του MQTT ελέγχει αν είναι το σύστημα ασφαλείας είναι ενεργό, πράγμα που συμβαίνει όταν το secFlag είναι true. Στην συνέχεια ελέγχει αν το κύκλωμα είναι κλειστό. Στην περίπτωση είναι, δηλαδή η μεταβλητή opened είναι true και το pin είναι HIGH κοινοποιεί στον server την συμβολοσειρά OPEN και θέτει το open = false και το closed = true. Στην περίπτωση που ο μαγνήτης πάνω στην πόρτα πλησιάσει τον διακόπτη το κύκλωμα θα ανοίξει και η κατάσταση στο pin θα γυρίσει σε LOW και πλέον καθώς η μεταβλητή closed είναι true θα δημοσιεύσει στον server την συμβολοσειρά CLOSED και θα γυρίσει την μεταβλητή close σε false και την open σε true.

Ακολουθεί η συνάρτηση void messageReceived() στην οποία, όταν ο μικροελεγκτής λάβει μήνυμα από τον server για την κατάσταση της ασφάλειας, θα γίνουν οι κατάλληλες ενέργειες για την ενεργοποίηση και απενεργοποίηση της ασφάλειας.

```
void messageReceived(String &topic, String &payload) {
  String msg = payload;

  if (msg == "secON") {
    digitalWrite(led, HIGH);
    secFlag = true;
    delay(2000);
  }

  else if (msg == "secOFF") {
    digitalWrite(led, LOW);
    secFlag = false;
    delay(250);
  }
}
```

Προγραμματισμός Συστήματος Ελέγχου Φωτιστικών με Dimmer

Σε αυτό το μέρος, πριν αναλύσουμε τον κώδικα θα αναφερθούμε στην προσέγγιση του προβλήματος πρώτα θεωρητικά. Στην Ελλάδα η συχνότητα του ρεύματος από την ΔΕΗ είναι στα 50Hz. Αυτό σημαίνει ότι έχουμε 50 ημιτονοειδή κύματα το δευτερόλεπτο, άρα κάθε ημιτονοειδές κύμα κρατάει για T = 1000ms / 50 = 20ms. Καθώς υπάρχουν δύο κορυφές στο ημίτονο, εύκολα καταλαβαίνουμε ότι η συνάρτηση περνάει από το 0 κάθε 10ms. Αυτό σημαίνει ότι πρέπει να βρούμε ένα τρόπο να ενεργοποιούμε το TRIAC μέσα σε αυτό το διάστημα για να παίρνουμε το επίπεδο φωτισμού που θέλουμε.

Αν ενεργοποιήσουμε αμέσως το TRIAC μετά το zero crossing τότε η λάμπα θα φωτιστεί 100%, αν περιμένουμε στο τέλος των 10ms τότε δεν θα φωτιστεί καθόλου, καθώς θα υπάρξει αμέσως zero crossing και θα επαναληφθεί η διαδικασία και αν το ενεργοποιήσουμε στα 5ms τότε η λάμπα θα φωτιστεί κατά το ήμισυ. Αυτό που πρέπει να κάνουμε είναι να γράψουμε ένα πρόγραμμα που θα αποφασίζει πότε θα ενεργοποιεί το TRIAC ανάλογα με το επίπεδο dimming που θέλουμε.

Ένας τρόπος να το κάνουμε είναι με την συνεχή «σφυγομέτρηση» του pin που μας ειδοποιεί το κύκλωμα για zero crossing. Αυτή η μέθοδος βέβαια δεν μας δίνει ευελιξία στον μικροελεγκτή καθώς θα είναι αναγκασμένος να ασχολείται μόνο με αυτό. Για ένα μικροελεγκτή που έχει και άλλες συναρτήσεις να εκτελέσει όπως αυτός στην εφαρμογή μας Χρειάζεται άλλη μέθοδο για να εκτελεί αυτά που πρέπει όταν υπάρχει zero crossing. Το πρόβλημα αυτό το λύνουμε χρησιμοποιώντας τις συναρτήσεις interrupt. Μία interrupt συνάρτηση μένει ανενεργή μέχρι να συμβεί ένα γεγονός που την αφορά και τότε διακόπτει το πρόγραμμα εκτελεί τις εντολές τις και επιστρέφει ακριβώς στο σημείο το οποίο σταμάτησε και συνεχίζεται η void loop(). Στην παρούσα εφαρμογή, λοιπό, μία interrupt συνάρτηση παρακολουθεί το zc pin και αν αυτό επιστρέψει HIGH, τότε διακόπτεται το πρόγραμμα και εκτελείται η συνάρτηση.

Μόλις Το πρόγραμμα πάει στην interrupt συνάρτηση, πρέπει να υπολογιστεί ο χρόνος που χρειάζεται να περιμένει το TRIAC πριν ενεργοποιηθεί. Στο πρόγραμμα το επίπεδο του dimming αποθηκεύεται σε ένα byte και η τιμή του είναι από 0 έως 255. Οπότε μπορούμε να χωρίσουμε τα 10ms σε βήματα. Κάθε βήμα είναι 1000 / 255 = 35μs (περίπου). Άρα πολλαπλασιάζουμε την τιμή του byte με 35μs και το αποτέλεσμα είναι ο χρόνος που θα χρειάζεται το να περιμένουμε πριν ο μικροελεγκτής ενεργοποιήσει το TRIAC.

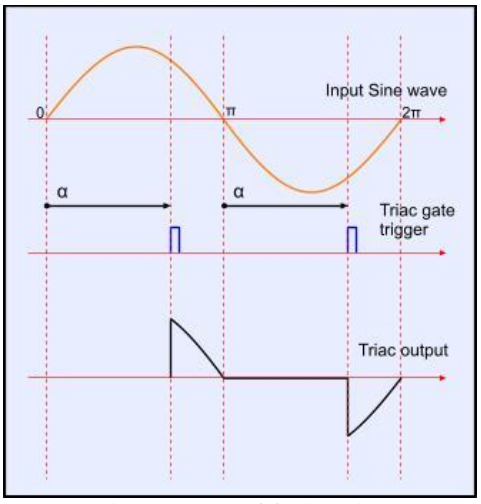

Figure 4-33 TRIAC triggering

Επιπλέον interruptions θα χρησιμοποιηθούνε από την "OneButton.h" για την λειτουργικότητα των πραγματικών διακοπτών. Τώρα μένει μόνο να παρουσιάσουμε και να αναλύσουμε τον κώδικα.

Όπως και για το σύστημα ασφαλείας, ο μικροελεγκτής που θα χρησιμοποιήσουμε είναι ο ESP8266 – 12F και θα προσπεράσουμε τα σημεία του κώδικα που είναι κοινά και θα επικεντρωθούμε στην λειτουργεία του dimmer. Ο κώδικας φαίνεται ολόκληρος στο παράρτημα μαζί με τους προηγούμενους. Η επιπλέον βιβλιοθήκη και τα topics που ορίσαμε για το dimmer είναι:

```
#include "OneButton.h"
```

```
char* lightsTopic = "MyHome/lights/LVR_Light";
char* dimButtonsTopic = "MyHome/lights/DMB_Light";
```

Το ένα είναι για την αποδοχή του επιπέδου dimming από τον server που κυμαίνεται από 0 έως 100 και το άλλο είναι για την αποστολή στον server το επίπεδο dimming που ορίζεται από τους πραγματικούς διακόπτες. Έπειτα δηλώνουμε τις μεταβλητές που εξυπηρετούν τα pins. Το AC_pin αποτελεί το output που δίνουμε στο κύκλωμα για να ενεργοποιήσει το TRIAC και το zc αποτελεί το input που ενημερώνει το πρόγραμμα πότε υπάρχει zero crossing.

```
//Light Circuit
int AC_pin = 12;
int zc = 13;
byte dim = 0;

//Button
OneButton increase(16, false);
OneButton decrease(4, false);
int maxInc;
int minDec;
```

Στην void setup() δηλώνουμε τον ρόλο τους και την αρχική τους κατάσταση. Σε αυτή την συνάρτηση συνήθως δηλώνονται και τα interruptions (διακοπές) αλλά με τον τρόπο που έχουμε εφαρμόσει το OTA δεν βόλευε να τα δηλώσουμε εκεί οπότε τα δηλώσαμε στην void loop(). Όπως και στο σύστημα ασφαλείας, έχουμε βάλει τον μικροελεγχτή να περιμένει 15 δευτερόλεπτα κατά το άνοιγμά του σε περίπτωση που λάβει κάποια ενημέρωση, και σε αυτή την εφαρμογή. Αν τα interruptions είχαν δηλωθεί νωρίτερα στην void setup(), θα διακοπτόταν το OTA με το zero crossing συνεχώς, με αποτέλεσμα να ήταν αδύνατον να εφαρμόσουμε την ενημέρωση.

```
pinMode(AC_pin, OUTPUT);
pinMode(zc, INPUT_PULLUP);
```

Αυτό μας οδηγεί στην void loop που λαμβάνει χώρα η OTA ενημέρωση, δηλώνονται τα interruptions και καλούνται οι συναρτήσεις για τον ορισμό του dimming είτε από μήνυμα του server είτε από τους πραγματικούς διακόπτες. Η void loop έχει την εξής μορφή.

```
void loop() {
  if (ota_flag) {
    while (time_elapsed < 15000) {
      ArduinoOTA.handle();
```

```
      time_elapsed = millis();
      delay(10);
    }
    ota_flag = false;

    attachInterrupt(zc, light, FALLING);
    increase.attachClick(clickA);
    increase.attachPress(pressA);
    increase.attachDoubleClick(doubleClick);
    decrease.attachClick(clickB);
    decrease.attachPress(pressB);
    decrease.attachDoubleClick(doubleClick);
  }

  if (!client.connected())
    connect();

  client.loop();

  /*-----------------BUTTONS-----------------*/
  increase.tick();
  decrease.tick();
  /*-----------------------------------------*/
}
```

Με το πέρας, λοιπόν, των 15 δευτερολέπτων που περιμένει ο μικροελεγκτής την ενημέρωση δηλώνονται τα interruptions. Με την χρήση της OneButton.h, στο πάτημα των pushbuttons γίνεται διακοπή του προγράμματος και ανάλογα με το ποιο κουμπί πατήθηκε αυξάνει ή μειώνει την ένταση της φωτεινότητας σύμφωνα με τον τρόπο που πατήθηκε αυτό το κουμπί. Αν πατήθηκε το κουμπί αύξησης φωτεινότητας στιγμιαία τότε το increase interruption καλεί η συνάρτηση clickA, αν πατήθηκε παρατεταμένα, τότε καλεί η pressA και αν πατήθηκε δύο φορές γρήγορα, τότε καλεί η doubleClick. Αν πατήσουμε το κουμπί μείωσης της έντασης τότε καλούνται οι αντίστοιχες συναρτήσεις από το decrease interruption όπως περιγράφονται παραπάνω.

Με την συνάρτηση clickA το byte για το dimming αυξάνει +10. Με την συνάρτηση pressA το byte γίνεται 255 και τέλος με την συνάρτηση doubleClick το byte γίνεται 128. Για τις decrease συναρτήσεις ισχύει ακριβώς το αντίθετο. Δηλαδή έχουμε -10 στην clickB και στη pressABτο byte γίνεται 0. Οι συναρτήσεις έχουν ως εξής:

```
void clickA() {
  float x;
  int percentage;
  if (dim <= 245) {
    dim += 10;
  }
  else if (dim > 250) {
    dim = 255;
  }

  x = (dim * 100) / 255;
  percentage = int(x);
  sprintf(lightMsg, "%d", percentage);
```

```
  client.publish(dimButtonsTopic, lightMsg);
}

void clickB() {
  float x;
  int percentage;
  if (dim >= 10) {
    dim -= 10;
  }
  else if (dim < 10) {
    dim = 0;
  }

  x = (dim * 100) / 255;
  percentage = int(x);
  sprintf(lightMsg, "%d", percentage);
  client.publish(dimButtonsTopic, lightMsg);
}

void pressA() {
  dim = 255;
  int x = 100;

  delay(5);

  sprintf(lightMsg, "%d", x);
  client.publish(dimButtonsTopic, lightMsg);
}

void pressB() {
  dim = 0;
  int x = 0;

  delay(5);

  sprintf(lightMsg, "%d", x);
  client.publish(dimButtonsTopic, lightMsg);
}

void doubleClick() {
  dim = 128;
  int x = 50;

  delay(5);

  sprintf(lightMsg, "%d", x);
  client.publish(dimButtonsTopic, lightMsg);
}
```

Οι συναρτήσεις αυτές καθορίζουν την μεταβλητή dim η οποία χρησιμοποιείται για την αυξομείωση του επιπέδου dimming στην λάμπα μας. Στην ουσία αυτή η μεταβλητή αποτελεί ο βασικός παράγοντας για τον καθορισμό του input στο κύκλωμα. Η τιμή αυτή όμως είναι σε

byte, δηλαδή, κυμαίνεται μεταξύ του 0 έως 255, οπότε χρειάζεται η κατάλληλη μετατροπή για να κυμαίνεται μεταξύ του 0 και του 100 όταν θα σταλεί στον server για ενημέρωση. Η ενημέρωση του μικροελεγκτή, όμως, από τον server συμβαίνει μέσω της messageReceived όπως και στις προηγούμενες εφαρμογές. Οπότε η συνάρτηση αυτή έχει την εξής μορφή:

```
void messageReceived(String &topic, String &payload) {
  String msg = payload;
  int fdim;
  float multiplier, percentage;

  percentage = msg.toInt();
  multiplier = percentage / 100;
  fdim = (int)255 * multiplier;

  dim = (byte)fdim;
}
```

Όπως και στις προηγούμενες εφαρμογές λοιπόν καλείτε η συνάρτηση messageReceived κάθε φορά που υπάρχει MQTT μήνυμα από τον server και η συνάρτηση στη συγκεκριμένη περίπτωση τροποποιεί την μεταβλητή dim ώστε να μπορούμε να την χρησιμοποιήσουμε στο πρόγραμμα παρακάτω. Όπως είπαμε ο server έχει κλίμακα από 0 έως 100, αλλά επειδή ο μικροελεγκτής αποθηκεύει το επίπεδο dimming σε μία byte μεταβλητή, κάνουμε την απαραίτητη μετατροπή.

Τέλος, μας έχει μείνει η συνάρτηση void light(), η οποία είναι κα η συνάρτηση η οποία ξεκινάει να δουλεύει όταν εντοπιστεί zero crossing. Όπως είπαμε και παραπάνω στο θεωρητικό μέρος, παίρνουμε την τιμή του dim και σύμφωνα με το βήμα που είναι 35μs βρίσκουμε τον χρόνο που θα περιμένει ο μικροελεγκτής μετά το zero crossing για να ενεργοποιήσει το TRIAC. Στις περιπτώσεις που το byte είναι 0 ή 255 τότε δεν το ενεργοποιούμε καθόλου ή το ενεργοποιούμε κατ' ευθείαν, αντίστοιχα, χωρίς να χρειάζεται να υπολογίσουμε χρόνο αναμονής.

```
void light() {
  if (dim < 1) {
    digitalWrite(AC_pin, LOW);
  }

  if (dim > 254) {
    digitalWrite(AC_pin, HIGH);
  }

  if (dim > 0 && dim < 255) {
    delayMicroseconds(37 * (255 - dim));
    digitalWrite(AC_pin, HIGH);
    delayMicroseconds(15);
    digitalWrite(AC_pin, LOW);
  }
}
```

4.4.4 Διαμόρφωση Configuration Files

Μέχρι αυτό το σημείο, έχουμε στήσει τον MQTT διακομιστή, έχουμε φτιάξει τα κυκλώματα και έχουμε προγραμματίσει τους μικροελεγκτές για την διαχείριση του οικιακού εξοπλισμού. Σε αυτό το υποκεφάλαιο θα δημιουργήσουμε στον server τα items που καθιστούν δυνατή την εικονική διαχείριση του εξοπλισμού μέσω των μικροελεγκτών, καθώς και τα items που μας δίνουν δεδομένα καιρικά και αστρονομικά των οποίων οι μεταβλητές θα μπορούν να χρησιμοποιηθούν για τον προγραμματισμό της μηχανής κανόνων για λόγους αυτοματοποίησης. Έπειτα θα δημιουργήσουμε το sitemap για να έχουμε την επιθυμητή διεπαφή μέσω της οποίας θα χειριζόμαστε το έξυπνο σπίτι και τέλος θα προγραμματίσουμε την μηχανή κανόνων για την αυτοματοποίηση των συσκευών.

Όλα τα αρχεία διαμόρφωσης που μας ενδιαφέρουν μπορούμε να τα προσεγγίσουμε στον server με την χρήση ssh και τα αντίστοιχα directories βρίσκονται στο directory /etc/openhab2/. Για αρχή θα ξεκινήσουμε με την δημιουργία των item files και things files που θα περιγράφουν την φύση των αντικειμένων μας το κανάλι επικοινωνίας και τους συνδέσμους μεταξύ τους.

Items File

Στο directory /etc/openhab2/items θα δημιουργήσουμε ένα αρχείο με όνομα home.items χρησιμοποιώντας την εντολή sudo nano home.items. Για την διαχείριση των μικροελεγκτών που προγραμματίσαμε προηγουμένως θα δηλώσουμε συνολικά 9 items. Ξεκινώντας με το σύστημα διαχείρισης με ρελέ, δηλώνουμε ένα item το οποίο αποτελεί ο εικονικός διακόπτης για την διαχείριση του ρελέ. Η δήλωση είναι η εξής:

```
Switch MBR_light "Bedroom Ceiling Light" <light> (gMyroom, Lights, gLights)
                                        {mqtt=
                                      ">[broker:MyHome/lights/MBR_Light:command:ON:L1ON],
                                     >[broker:MyHome/lights/MBR_Light:command:OFF:L1OFF],
                                   <[broker:MyHome/lights/MBR_Light:state:MAP(onoff.map)]"}
```

Όπως προαναφέραμε στο κεφάλαιο 3, στην αρχή δηλώνουμε την λειτουργεία του item, διακόπτης, στην συνέχεια δηλώνουμε το όνομα που θα χρησιμοποιούμε για τον προγραμματισμό της μηχανής κανόνων και την διεπαφή, MBR_light, στη συνέχεια δηλώνουμε το όνομα με το οποίο θα εμφανίζεται ο διακόπτης στην διεπαφή και έπειτα στις παρενθέσεις βάζουμε τα group στα οποία θα ανήκει. Στην διεπαφή τα group φαίνονται σαν μία οντότητα στη οποία όταν κάνουμε κλικ, τότε θα εμφανιστούν τα items τα οποία περιέχουν. Τα groups χρησιμοποιούνται για την οργάνωση της διεπαφής. Μέσα στις αγκύλες δηλώνουμε το κανάλι επικοινωνίας και τους συνδέσμους. Το σύμβολο "<" πριν το topic δηλώνει ότι από κει παίρνει εισερχόμενα MQTT μηνύματα και με το αντίστροφο σύμβολο ότι σε αυτό το topic στέλνει MQTT μηνύματα. Στις δύο πρώτες προτάσεις περιγράφεται πως αν εικονικός διακόπτης γίνει ON, το μήνυμα που θα σταλεί στον μικροελεγκτή θα είναι το L1ON, και αντίστοιχα L1OFF αν ο διακόπτης γίνει OFF. Ο μικροελεγκτής έχει προγραμματιστεί κατάλληλα για την επεξεργασία αυτού του μηνύματος. Η τελευταία πρόταση υπάρχει για το ενδεχόμενο που αλλάξει η κατάσταση του ρελέ από τον πραγματικό διακόπτη. Σε αυτή την περίπτωση ο μικροελεγκτής στέλνει MQTT μήνυμα την κατάσταση του διακόπτη 0 ή 1 και με την χρήση του αρχείου onoff.map που βρίσκεται στο directory /etc/openhab2/transform αυτό μεταφράζεται σε ON ή OFF που γίνεται κατανοητό από τον εικονικό διακόπτη και παίρνει την αντίστοιχη κατάσταση. Το αρχείο onoff.map έχει την εξής μορφή:

```
0=OFF
1=ON
```

Στην συνέχεια δημιουργούμε τον διακόπτη με τον οποίο θα ενεργοποιούμε την ασφάλεια. Η δήλωσή του είναι η εξής:

```
Switch SecuritySys "Security" <siren> {mqtt=
">[broker:MyHome/security/Main_Door:command:ON:secON],
>[broker:MyHome/security/Main_Door:command:OFF:secOFF]"}
```

Το MQTT μήνυμα σε αυτόν τον διακόπτη αφορά την λειτουργεία του LED για την υπόδειξη ότι το σύστημα ασφαλείας είναι ενεργό. Το υπόλοιπο σύστημα είναι προγραμματισμένο στην μηχανή κανόνων που θα δούμε παρακάτω. Έπειτα δημιουργούμε ένα item τύπου contact που μας υποδηλώνει την λειτουργεία του μαγνητικού διακόπτη.

```
Contact MainDoorSensor "Main Door [%s]" <frontdoor> (Doors, gLiving)
{mqtt="<[broker:MyHome/doors/Main_Door:state:default]"}
```

Με την χρήση του [%s] στο όνομα του item, θα εμφανίζεται και η κατάστασή του. Η κατάσταση έρχεται από το κανάλι επικοινωνίας και με την δήλωση state:default κρατάει ο server ακριβώς το string που έλαβε.

Συνεχίζουμε με την δήλωση των items που αφορούν το dimmer. Για την διαχείρισή του χρησιμοποιούμε δύο items. Το ένα item είναι τύπου dimmer, δηλαδή εμφανίζεται σαν slider και στέλνει τιμές από 0 έως 100. Η δήλωση αυτουνού είναι η εξής:

```
Dimmer LVR_light "Living Room Dimmer" <slider> (gLiving)
{mqtt=
">[broker:MyHome/lights/LVR_Light:state:*:default],
<[broker:MyHome/lights/DMB_Light:state:default]"}
```

Στο κανάλι επικοινωνίας η σύνδεση γίνεται όπως με τον μαγνητικό διακόπτη στα εισερχόμενα μηνύματα και στα εξερχόμενα είναι αναγκαία η προσθήκη του συμβόλου “*” μεταξύ state και default για να υποδηλώσουμε ότι το εξερχόμενο μήνυμα μπορεί να είναι οποιαδήποτε συμβολοσειρά. Με αυτόν τον τρόπο λαμβάνουμε και στέλνουμε δεδομένα στον μικροελεγκτή που ελέγχει το dimmer. Στο ενδεχόμενο που χρησιμοποιήσουμε τους πραγματικούς διακόπτες πρέπει να ενημερωθεί και ο server να προσαρμοστεί στις αλλαγές αυτές. Γι’ αυτό τον λόγο δημιουργούμε ένα ψεύτικο εικονικό διακόπτη ο οποίος έχει προγραμματιστεί στην μηχανή κανόνων να είναι OFF ή ON στο 0 και 100 του slider αντίστοιχα αλλά και το slider να συμπεριφέρεται αντίστοιχα με την κατάσταση του διακόπτη. Αυτός ο διακόπτης είναι χρήσιμος και για τα active groups που θα εξηγήσουμε παρακάτω. Ο διακόπτης έχει την εξής δήλωση:

```
Switch Dummy1 "Living Room Dimmer" <light> (gLights, Lights)
```

Δεν χρειάζεται κάποιο κανάλι επικοινωνίας καθώς όπως είπα είναι ψεύτικος και η όλες οι διαδικασίες του ορίζονται στην μηχανή κανόνων.

Μέχρι αυτό το σημείο έχουμε δημιουργήσει τις κατάλληλες οντότητες για την διαχείριση του εξοπλισμού που συνδέεται με τους μικροελεγκτές μας. Ωστόσο έχουμε προγραμματίσει τα MCU να μοιράζονται με τον server το RSSI τους. Αυτό μας είναι χρήσιμο καθώς αποτελεί ένα καλό ξεκίνημα για εντοπισμό προβλημάτων σε περίπτωση που το σύστημά μας δεν ανταποκρίνεται όπως θα θέλαμε. Τα items που αντιπροσωπεύουν αυτές τις τιμές στον server είναι τα εξής:

```
Number MBR_BCL_RSSI "Ceiling Light [%d/4]" <signal> (gMyroom_RSSI)
{mqtt="<[broker:MyHome/rssiLights/MBR_Light:state:default]"}
```

```
String RSSI_Current_MBR_BCL "[MAP(rssi.map):%s]" <number>
                            (gMyroom_RSSI)
{mqtt="<[broker:MyHome/rssiLights/MBR_Light:state:default]"}

Number LR_HMD_RSSI "Main door [%d/4]" <signal (gLiving_RSSI)
 {mqtt="<[broker:MyHome/rssiDoors/Main_Door:state:default]"}

String RSSI_Current "[MAP(rssi.map):%s]" <number> (gLiving_RSSI)
 {mqtt="<[broker:MyHome/rssiDoors/Main_Door:state:default]"}

 Number LR_LCL_RSSI "Dimmer [%d/4]" <signal> (gLiving_RSSI)
{mqtt="<[broker:MyHome/rssiLights/LVR_Light:state:default]"}
```

Για κάθε μικροελεγκτή έχουμε δύο items. Το ένα είναι ο αριθμός που έρχεται από τον μικροελεγκή και μας παρουσιάζεται με την χρήση του δυναμικού εικονιδίου <signal> και το άλλο είναι μία συμβολοσειρά η οποία μετά την αντιστοίχιση του αριθμού που έρχεται από τον server με την συμβολοσειρά μέσα στο αρχείο rssi.map μας εμφανίζει την συμβολοσειρά το αρχείο rssi.map έχει την εξής μορφή:

```
 0=No Connection
 1=Bad Conncetion
2=Medium Connection
 3=Good Connection
4=Strong Connection
```

Για την ομαδική διαχείριση και επίβλεψη του εξοπλισμού δημιουργούμε τα λεγόμενα active groups. Τα active groups είναι διαδραστικά groups, πράγμα που σημαίνει ότι μπορούμε να επέμβουμε σε όλα τα items ταυτόχρονα απλά με την χρήση ενός group. Για παράδειγμα το active group lights είναι για να κλείνουν όλα τα φώτα του σπιτιού που τυχαίνει να είναι ανοικτά με την χρήση ενός και μόνο διακόπτη. Παράλληλα αν μπούμε μέσα στο group, μας υποδικνύει ποιες οντότητες είναι ON και ποιες OFF. Το active group doors μας υποδηκνύει πόσες πόρτες στο σύστημά μας είναι ανοικτές και αν μπούμε μέσα στο group μπορούμε να δούμε ποιες ακριβώς είναι. Τα active groups που χρησιμοποιούμε σε αυτό το σύστημα δηλώνονται ως εξής:

```
        Group:Switch:OR(ON, OFF) Lights "All Lights [(%d)]"
Group:Contact:OR(OPEN, CLOSED) Doors "Open Doors [(%d)]"    <door>
```

Το πρώτο είναι ένα group τύπου διακόπτη και μας απαριθμεί τα ενεργά φώτα και αν έστω και ένα είναι ενεργό τότε ο διακόπτης του group είναι στην κατάσταση ON. Το ίδιο συμβαίνει και για το δεύτερο group που μας απαριθμεί τις ανοικτές πόρτες. Δεν έχει βέβαια διακόπτη, αλλά αν έστω και μία πόρτα είναι ανοικτή τότε η κατάσταση του δυναμικού εικονιδίου είναι αντίστοιχη. Αυτές οι ομαδοποιήσεις γενικότερες καθιστούν ένα αρκετά μεγάλο σύστημα έξυπνου σπιτιού πιο εύκολο στην διαχείρισή του.

Σε αυτό το σημείο έχουμε δημιουργήσει όλες τις οντότητες για την αποστολή και λήψη MQTT μνημάτων από τους μικροελεγκτές μας. Παρ' όλα αυτά, έχουμε επιπλέον items στην διάθεσή μας που αφορούν καιρικά δεδομένα, αστρονομικά καθώς και την τρέχουσα ώρα και ημερομηνία. Αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν σαν inputs στην μηχανή κανόνων του openHAB για την διαχείριση οικιακών συσκευών. Για παράδειγμα μπορούμε να προγραμματίσουμε μία συσκευή να κλείνει και να ανοίγει ανάλογα με την ώρα της ημέρας ή στην περίπτωση βροχής να απενεργοποιείται το σύστημα ποτίσματος, αν αυτό είναι ενεργό. Γενικότερα, η χρήση δεδομένων σαν και αυτά σε ένα έξυπνο σύστημα μας φέρνει πιο κοντά σε ένα πλήρες αυτοματοποιημένο σύστημα οικιακού αυτοματισμού.

Για να πάρουμε τα καιρικά δεδομένα πρέπει πρώτα να εγκαταστήσουμε στο PaperUI το weather binding. Έπειτα στο directory /etc/openhab2/services βρίσκουμε το weather.cfg configuration file και φτιάχνουμε τις ρυθμίσεις για τα δεδομένα. Αυτές είναι οι εξής:

```
apikey.ForecastIo=37ee26f92c1aa05e2ef1627692251c00
location.home.name=home
location.home.latitude=35.334
location.home.longitude=25.1329
location.home.provider=ForecastIo
location.home.language=en
location.home.updateInterval=10
```

Στην συγκεκριμένη εργασία επιλέγουμε το ForecastIO για πάροχο των δεδομένων. Για να μπορούμε να χρησιμοποιήσουμε τα δεδομένα αυτού του παρόχου πρέπει να κάνουμε εγγραφή στον πάροχο με το apikey που μας δίνει. Έπειτα το δηλώνουμε στις ρυθμίσεις στο weather.cfg. Επιπλέον δηλώνουμε ένα όνομα για το σύστημα που αφορούν τα δεδομένα καθώς και τις γεωγραφικές συντεταγμένες της πόλης που μας αφορά. Τέλος δηλώνουμε την γλώσσα που θα μας εμφανίζονται τα δεδομένα καθώς και τον χρόνο ενημέρωσης.

Έπειτα, μπορούμε να δηλώσουμε τις οντότητες των δεδομένων στο items file. Η δήλωσή τους είναι η εξής:

```
Number Temperature "Temperature [%.2f °C]" <temperature> (gWeather)
  {weather="locationId=home, type=temperature, property=current"}

Number Temperature_Feel "Temperature feel [%.2f °C]" <temperature>
                        (gWeather)
   {weather="locationId=home, type=temperature, property=feel"}

    Number Humidity "Humidity [%d %%]" <humidity> (gWeather)
 {weather="locationId=home, type=atmosphere, property=humidity"}

   Number Pressure "Pressure [%.2f mb]" <pressure> (gWeather)
 {weather="locationId=home, type=atmosphere, property=pressure"}

      Number Wind "Wind Speed [%.2f km/h]" <wind> (gWeather)
      {weather="locationId=home, type=wind, property=speed"}

         Number Rain "Rain [%.2f] mm/h" <rain> (gWeather)
 {weather="locationId=home, type=precipitation, property=rain"}
```

Αυτές οι οντότητες αντιπροσωπεύουν δεδομένα που αφορούν την θερμοκρασία, την αισθητή θερμοκρασία, την υγρασία, την ατμοσφαιρική πίεση, την ταχύτητα του ανέμου και τα χιλιοστά βροχής. Υπάρχουν και άλλα δεδομένα που μπορούμε να προσθέσουμε με τον ίδιο τρόπο. Όλα αυτά τα έχουμε ομαδοποιήσει στο group gWeather.

Για τα αστρονομικά δεδομένα εγκαθιστούμε το astro binding και να δημιουργήσουμε δύο αντικείμενα στο things file. Το ένα θα αφορά τα δεδομένα που έχουν σχέση με τον ήλιο και το άλλο τα δεδομένα που έχουν σχέση με το φεγγάρι. Επίσης εκεί δίνουμε και την ετικέτα της γεωγραφικής θέσης του συστήματος κα τον χρόνο ανανέωσης των δεδομένων. Το things file θα είναι έτσι:

```
 astro:sun:home [ geolocation="35.338735,25.144213", interval=60 ]
astro:moon:home [ geolocation="35.338735,25.144213", interval=60 ]
```

Στο items file δηλώνουμε τα δεδομένα που θέλουμε να μας εμφανίζονται. Στην δικιά μας εφαρμογή αυτά είναι η ώρα ανατολής του ηλίου, η ώρα δύσης του ηλίου, το αζιμούθιο του ήλιου και η ανύψωσή του, η φάση της σελήνης, καθώς και η ανύψωσή της και το αζιμούθιο. Το items file θα έχει την εξής μορφή:

```
Number Sun_Elevation "Sun Elevation" <sun> (gAstronomy)
    { channel = "astro:sun:home:position#elevation" }

Number Sun_Azimuth "Sun Azimuth" <sun> (gAstronomy)
    { channel = "astro:sun:home:position#azimuth" }

DateTime Sunrise_Time "Sunrise [%1$tH:%1$tM]" <sunrise>
                    (gAstronomy)
        { channel = "astro:sun:home:rise#start" }
  DateTime Sunset_Time "Sunset [%1$tH:%1$tM]" <sunset>
                    (gAstronomy)
          { channel = "astro:sun:home:set#start" }

Number Moon_Elevation "Moon Elevation" <moon> (gAstronomy)
    { channel = "astro:moon:home:position#elevation" }

Number Moon_Azimuth "Moon Azimuth" <moon> (gAstronomy)
    { channel = "astro:moon:home:position#azimuth" }

String Moon_Phase "Moon Phase [MAP(astro.map):%s]" <moon>
                    (gAstronomy)
        { channel = "astro:moon:home:phase#name" }
```

Τα δεδομένα που αφορούν την φάση της σελήνης δεν είναι άμεσα κατανοητά οπότε χρησιμοποιούμε το αρχείο astro.map στο /etc/openhab2/transform directory για την απόδοσή τους στα αγγλικά. Το αρχείο έχει την εξής μορφή:

```
                NEW=Newmoon
WAXING_CRESCENT=Increasing Crescent
     FIRST_QUARTER=First Quarter
   WAXING_GIBBOUS=Increasing Moon
              FULL=Fullmoon
   WANING_GIBBOUS=Decreasing Moon
     THIRD_QUARTER=Last Quarter
WANING_CRESCENT=Decreasing Crescent
```

Όπως και τα καιρικά δεδομένα, έτσι και τα αστρονομικά τα έχουμε βάλει σε ένα group για την ευκολότερη διαχείρισή τους στην διεπαφή. Τέλος, για την εμφάνιση των δεδομένων που αφορούν την ώρα και την ημερομηνία δημιουργούμε άλλο ένα thing, το οποίο έχει την εξής μορφή:

```
ntp:ntp:home [ hostname="nl.pool.ntp.org", refreshInterval=60,
                    refreshNtp=30 ]
```

Φυσικά όπως και για τα προηγούμενα πρέπει να εγκαταστήσουμε το κατάλληλο binding το οποίο είναι το NTP binding. Έπειτα στο items file δημιουργούμε τα δεδομένα που αντιπροσωπεύουν τα δεδομένα μας. Το items file έχει την εξής μορφή:

```
DateTime CurrentDate "Date [%1$tA, %1$td.%1$tm.%1$tY, %1$tR]"
        <calendar> { channel="ntp:ntp:home:dateTime" }
```

Τέλος, στο PaperUI υπάρχουν bindings για την διαχείριση smart TVs μέσω ων διεπαφών του openHAB. Κάποιος, με την επίσκεψη στο documentation του openHAB μπορεί να δει τα διαθέσιμα bindings για τις μάρκες κατασκευαστή των έξυπνων τηλεοράσεων. Με αυτό τον τρόπο αν στο σύστημα υπάρχει smart TV, τότε όχι μόνο είναι διαθέσιμος ο έλεγχός της από τις διεπαφές του openHAB από οπουδήποτε στον κόσμο, αλλά μπορεί και να ενταχθεί μέσα στην μηχανή κανόνων για ακόμα μεγαλύτερη αυτοματοποίηση του συστήματος.

Στην παρούσα εργασία η εφαρμογή έγινε πάνω σε μία Panasonic Viera smart TV οπότε χρειάστηκε και το αντίστοιχο binding, Panasonic Binding. Με την εγκατάσταση του binding αποκτούμε και το αρχείο panasonictv.cfg. Μέσα στο αρχείο αυτό βάζουμε την IP διεύθυνση της τηλεόρασης. Το αρχείο έχει την εξής μορφή:

```
Living_TV=192.168.1.121
```

Έπειτα στο items file δηλώνουμε τις οντότητες που μας ενδιαφέρουν. Οι εντολές για τα κουμπιά του πληκτρολογίου που θα χρησιμοποιηθούν στον σύνδεσμο του αντίστοιχου item βρίσκονται όλες στο documentation. Το items file έχει ως εξής:

```
Switch TVPOWER "Power" (gTV)
{ panasonictv=:"Living_TV:POWER" }

Switch VOLUMEUP "+" (gTV)
 { panasonictv=:"Living_TV:VOLUP" }

Switch VOLUMEDOWN "-" (gTV)
{ panasonictv=:"Living_TV:VOLDOWN" }

Switch TVINPUT "TV input" (gTV)
 { panasonictv=:"Living_TV:TV" }
```

Όλες οι οντότητες που αφορούν τον έλεγχο της τηλεόρασης ομαδοποιούνται μέσα στο group gTV. Μειονέκτημα της εφαρμογής αποτελεί ότι για ορισμένους κατασκευαστές όπως η Panasonic, διαχείριση είναι εφικτή μόνο με την χρήση διακοπτών, πράγμα το οποίο λειτουργεί μεν αλλά δεν είναι βολικό και επιπλέον, αν η τηλεόραση δεν έχει την δυνατότητα να κρατήσει το WiFi ανοικτό αφού τερματίσουμε την λειτουργεία της, τότε δεν μπορούμε να την ξαναενεργοποιήσουμε μέσω του openHAB, καθώς ο server μας επικοινωνεί μέσω του router κατευθείαν με την τηλεόραση.

Το items file, home.items βρίσκεται ολοκληρωμένο στο παράρτημα της πτυχιακής με σχόλια για την χρησιμότητα της κάθε οντότητας.

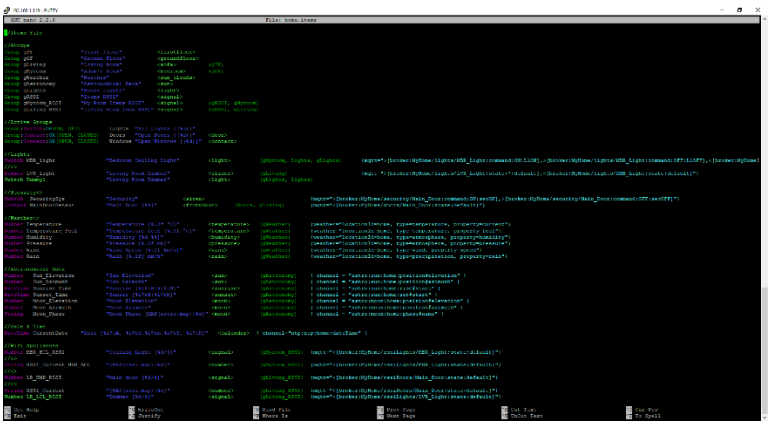

Figure 4-34 Items File

Sitemap File

Στο sitemap file δηλώνουμε τα items και τα groups που έχουμε για να φαίνονται στην διεπαφή του openHAB. Αν τα items βρίσκονται σε κάποιο group, τότε δεν χρειάζεται να το γράψουμε καθώς θα εμφανιστεί μέσα στο group. Εκτός και αν βέβαια θέλουμε να βρίσκεται και σε διαφορετικό σημείο της διεπαφής.

Το sitemap χωρίζεται από πλαίσια (frames) και για το κάθε ένα πρέπει να του δώσουμε κάποιο label, όπως πρέπει να δώσουμε label και στο sitemap το ίδιο. Για να εμφανιστούν τα items σωστά στο sitemap, πρέπει για κάθε ένα από αυτά να δηλώνουμε τον τύπο του και το ότι είναι item. Για παράδειγμα Text item=MainDoorSensor. Το sitemap της παρούσας εφαρμογής έχει την εξής μορφή:

```
sitemap home label="My Home"
{
        Frame label="Date & Time"
        {
                Text item=CurrentDate
        }
        Frame label="RSSI"
        {
                Group item=gRSSI
        }
        Frame label="House"
        {
                Group item=gFF label="First Floor" icon="firstfloor"
                Group item=gGF label="Ground Floor" icon="groundfloor"
                Group item=gLights label="House Lights" icon="light"
                Switch item=Lights mappings=[OFF="All off"]
                Group item=Doors
                Group item=Windows
        }
        Frame label="Security"
        {
                Switch item=SecuritySys
                Text item=MainDoorSensor
        }
        Frame label="Weather & Astronomical Data"
        {
                Text item=Temperature
                {
                        Frame label="Weather Data"
                        {
                                Text item=Temperature
                                Text item=Temperature_Feel
                                Text item=Humidity
                                Text item=Pressure
                                Text item=Wind
                                Text item=Rain
                        }
                }
                Group item=gAstronomy label="Astronomical Data" icon="sun"
        }
}
```

Rules File

Σε αυτό το μέρος θα εξηγηθεί ο τρόπος λειτουργείας της μηχανής κανόνων του openHAB. Αυτή η μηχανή προσφέρει διαδικασίες αυτοματισμού στο σύστημά μας οι οποίες ενεργοποιούνται από τις καταστάσεις των εισερχόμενων δεδομένων στον διακομιστή. Όπως αναφέραμε και στο κεφάλαιο 3, η σύνταξη της γλώσσας στη μηχανή κανόνων είναι βασισμένη στην Xbase, οπότε κατ' επέκταση, μοιράζεται πολλές λεπτομέρειες με την Xtend, η οποία είναι επίσης χτισμένη πάνω στην Xbase.

Πριν αναλύσουμε τον κώδικα της μηχανής, χρειάζεται να αναφέρουμε ένα action που χρειάστηκε να εγκατασταθεί για να έχει την δυνατότητα ο server να στέλνει e – mail. Το action αυτό είναι το Mail action. Για να έχουμε τις δυνατότητες του action πρέπει να καταχωρίσουμε τις κατάλληλες ρυθμίσεις στο configuration file. Οπότε, στο directory /etc/openhab2/services καταχωρούμε τα εξής:

```
hostname=smtp.gmail.com
port=587
username=adamfrag@gmail.com
password=********
from=adamfrag@gmail.com
tls=true
```

Από ότι βλέπουμε, καθορίζουμε το hostname του SMTP διακομιστή που χρησιμοποιούμε, καθώς και την πόρτα. Επίσης διευκρινίζουμε e – mail και κωδικό για τον διακομιστή σε περίπτωση που χρειάζεται πιστοποίηση για να καταχωρίσουμε τη διεύθυνση αποστολέα και τέλος καθορίζουμε και την διεύθυνση στην οποία θα λαμβάνουμε το mail. Στην δικιά μας την περίπτωση χρησιμοποιείται η ίδια. Η τελευταία μεταβλητή ορίζεται προαιρετικά σε true και χρειάζεται μόνο αν το STAARTTLS είναι ενεργοποιημένο για την σύνδεση. Μετά από αυτές τις ρυθμίσεις ο διακομιστής μας έχει την ευχέρεια να στέλνει mail.

Προχωρώντας παρακάτω, θα δούμε τον τρόπο λειτουργείας του συστήματος ασφαλείας. Γενικά ο αλγόριθμος λειτουργεί με τον εξής τρόπο: Όταν το item που εκφράζει τον αισθητήρα στην πόρτα λάβει ενημέρωση OPEN, τότε ελέγχει αν το switch item που εκφράζει την ασφάλεια είναι ON. Αν δεν είναι δεν γίνεται τίποτα. Αν είναι τότε στέλνει mail με τα στοιχεία που καθορίσαμε παραπάνω και η εφαρμογή στο κινητό στέλνει ειδοποίηση και ενημερωνόμαστε πως η πόρτα άνοιξε. Στον ίδιο αλγόριθμο έχει προστεθεί μία παραπάνω λειτουργεία αυτοματισμού η οποία έχει ως εξής: Στην περίπτωση που το σύστημα ασφαλείας είναι OFF και μαζί με αυτό το item που εκφράζει τον διακόπτη φωτός που βρίσκεται στο ίδιο δωμάτιο με τον αισθητήρα είναι OFF, τότε με το άνοιγμα της πόρτας ανάβει αυτόματα και το φως του δωματίου. Ο κώδικας είναι ο εξής

```
rule "Main Door lights/security"
        when
                Item MainDoorSensor received update OPEN
        then
                if(SecuritySys.state == ON)
                {
                        sendBroadcastNotification("Security is breached, somebody
                                                    opened the main door!")
                        sendMail("adamfrag@gmail.com","from openhab security","The
                                        security is breached
                }
                if(SecuritySys.state == OFF)
                {
                        if(MBR_light.state == OFF)
                        {
                                sendCommand(MBR_light, ON)
                        }
                }
end
```

Παρακάτω στο αρχείο των κανόνων περιγράφουμε τον τρόπο που θα λειτουργεί ο ψεύτικος εικονικός διακόπτης που προσθέσαμε για την λειτουργεία του dimmer. Ο πρώτος αλγόριθμος ελέγχει αν ο ψεύτικος διακόπτης παρέλαβα κάποια εντολή. Αν η εντολή αυτή είναι ON ΚΑΙ ταυτόχρονα το slider που ανταποκρίνεται στον μικροελεγκτή που ελέγχει το dimmer είναι στο 0, τότε το slider πάει στο 100. Αν η εντολή είναι OFF, τότε το slider πάει στο 0. Ο δεύτερος αλγόριθμος καθορίζει τον διακόπτη ανάλογα με την κατάσταση του slider. Αν αλλάξει η κατάσταση του slider, ελέγχουμε αν αυτή είναι διάφορη του μηδενός. Αν ναι, τότε ο ψεύτικος διακόπτης δέχεται εντολή να αλλάξει την κατάστασή του σε ON. Αν όχι, τότε δέχεται εντολή για να αλλάξει την κατάστασή του σε OFF.

```
//Dimmer
rule "Switch to Dimmer"
        when
                Item Dummy1 received command
        Then
                if(receivedCommand == ON && LVR_light.state == 0)
                {
                        sendCommand(LVR_light, 100)
                }
                else if(receivedCommand == OFF)
                {
                        sendCommand(LVR_light, 0)
                }
End

rule "Dimmer to Switch"
        when
                Item LVR_light changed
        Then
                if(LVR_light.state != 0)
                {
                        sendCommand(Dummy1, ON)
                } else {
                        sendCommand(Dummy1, OFF)
                }
end
```

Στην συνέχεια του rule file, βρίσκουμε κώδικα ο οποίος εφαρμόζει διαδικασίες αυτοματισμού οι οποίες ενεργοποιούνται από τα αστρονομικά δεδομένα που παίρνει ο server. Για παράδειγμα, ένας κανόνας ενεργοποιεί το φως του σπιτιού την ώρα που ο ήλιος δύει στην γεωγραφική μας τοποθεσία και ο άλλος το απενεργοποιεί την ώρα που ο ήλιος ανατέλλει. Προφανώς αυτές οι ώρες αλλάζουν με το πέρασμα του κύκλου του χρόνου γι' αυτό και χρησιμοποιούμαι τα astro data καθώς καθημερινά αλλάζουν. Ο κώδικας είναι ο εξής.

```
//Morning Switch
rule "Morning Switch OFF"
        when
                Channel 'astro:sun:home:rise$event' triggered START
        then
                if(MBR_light.state == ON)
                {
                        sendCommand(MBR_light, OFF)
                }
end
```

```
//Night Switch
rule "Night Switch ON"
        when
                Channel 'astro:sun:home:set#event' triggerd START
        then
                if(MBR_light.state == OFF)
                {
                        sendCommand(MBR_light, ON)
                }
end
```

Αυτοί οι κανόνες ελέγχουν πότε ξεκινάει η φάση ανατολής ή δύσης του ήλιου και εκείνη την στιγμή ενεργοποιούν τους κανόνες για να εφαρμόσει τους αντίστοιχους αυτοματισμούς.

5 Συνολική Λειτουργία

Σε αυτό το κεφάλαιο θα γίνει η επεξήγηση και η επίδειξη της λειτουργείας του συστήματος. Όπως προαναφέραμε, κεντρικός κόμβος του δικτύου είναι το Raspberry PI το οποίο τρέχει το λειτουργικό openHABian και έχει διαμορφωθεί κατάλληλα για να μας δίνει στις διεπαφές του openHAB τις αντίστοιχες απεικονίσεις των συσκευών που θέλουμε να διαχειριστούμε απομακρυσμένα. Επίσης υπάρχει και η μηχανή κανόνων για την αυτοματοποίηση του συστήματος ένα βήμα παραπάνω.

Ο server στέλνει MQTT μηνύματα, αντίστοιχα με την διάδρασή μας στην διεπαφή και οι μικροελεγκτές που διαχειρίζονται τις συσκευές μέσω των ηλεκτρονικών κυκλωμάτων εκτελούν τις αντίστοιχες εντολές για την εφαρμογή της κατάστασης της οποίας επιθυμούμε στην συσκευή. Η κατάσταση αυτή μπορεί να αποτελέσει ερέθισμα στην μηχανή κανόνων και να ενεργοποιηθεί η αλλαγή κατάστασης σε κάποιο άλλο αντικείμενο του συστήματος.

Οι μικροελεγκτές αντίστοιχα στέλνουν και αυτοί MQTT μηνύματα, αν είναι προγραμματισμένοι να το κάνουν, τα οποία ερμηνεύονται από τον server και ενημερώνονται οι αντίστοιχες καταστάσεις στην απεικόνιση των αντικειμένων.

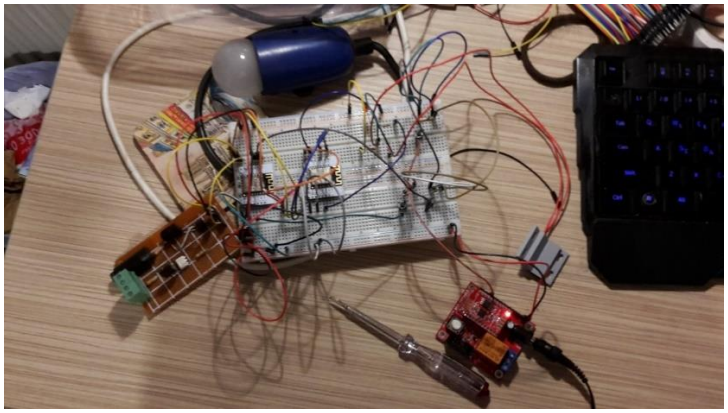

Figure 5-1 Κυκλώματα όλου του συστήματος

Οι διεπαφές του συστήματος στον υπολογιστή και στο smartphone, δομούνται όπως προαναφέραμε και στα προηγούμενα κεφάλαια με την διαμόρφωση του αρχείου sitemap στον εξυπηρετητή. Στη συγκεκριμένη υλοποίηση τα περισσότερα αντικείμενα βρίσκονται μέσα σε ομάδες για την καλύτερη οργάνωση της διεπαφής, οπότε δεν είναι άμεσα προσβάσιμα. Η απεικόνιση λοιπόν είναι η εξής:

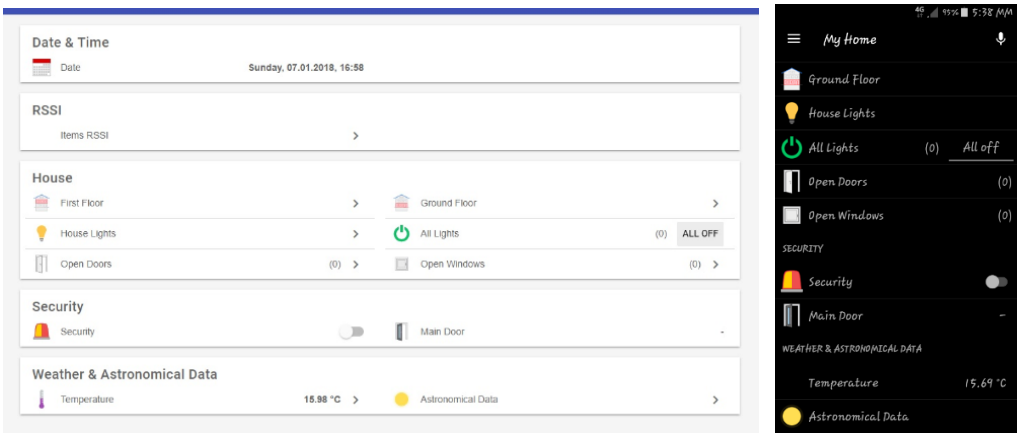

Figure 5-2 διεπαφή computer & smartphone

5.1 Σύστημα Ασφαλείας

Σε πρώτη φάση, θα αναλύσουμε και θα δούμε την λειτουργεία του συστήματος ασφαλείας. Με την χρήση του μαγνητικού διακόπτη, ο μικροελεγκτής στέλνει μήνυμα στον εξυπηρετητή που αφορά την κατάσταση της πόρτας. Αυτό είναι είτε OPEN ή CLOSED.

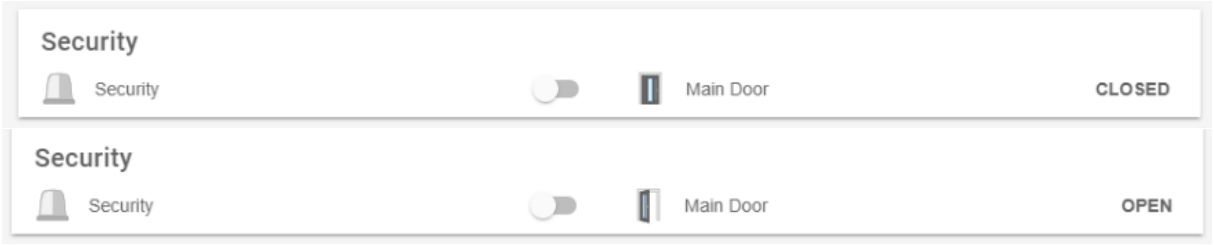

Figure 5-3 Security System

Στην περίπτωση που το σύστημα ασφάλειας είναι απενεργοποιημένο όπως παραπάνω, τότε αν το φως του δωματίου, που αντιστοιχεί στην πόρτα που είναι ο διακόπτης, είναι σβηστό, τότε με το άνοιγμα της πόρτας αυτό θα ανάψει. Αυτό γίνεται εφικτό με την χρήση της μηχανής κανόνων του openHAB.

Στην περίπτωση που το σύστημα ασφαλείας είναι ενεργοποιημένο, τότε στο ενδεχόμενο που η πόρτα ανοίξει αυτό θα ενημερώσει τους χρήστες του συστήματος με notification στο κινητό, καθώς και με e-mail για παραπάνω σιγουριά.

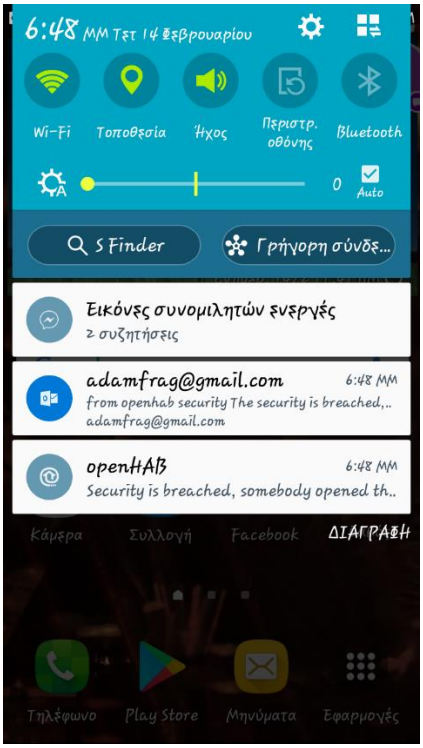

Figure 5-4 openhab notification and email

5.2 Dimmer

Για το dimmer, χρησιμοποιούμε ένα slider που στέλνει τιμές από το 1 έως το 100 στον μικροελεγκτή για να καθορίσει αυτός μετά την ένταση του φωτός. Για το dimmer έχει προστεθεί και ένας «ψεύτικος διακόπτης» ο οποίος όταν ενεργοποιείται κάνει την τιμή του slider 100 και όταν απενεργοποιείται την κάνει 0. Αυτό λειτουργεί και αντίθετα βέβαια, καθώς όταν το slider είναι στο 0, τότε ο διακόπτης απενεργοποιείται και όταν είναι σε οποιαδήποτε άλλη τιμή ενεργοποιείται. Επίσης και αυτό το κομμάτι είναι αποτέλεσμα της μηχανής κανόνων.

Στις παρακάτω φωτογραφίες απεικονίζεται η λάμπα η οποία ανάλογα με τον slider δίνει την μισή ισχύ, ολόκληρη την ισχύ ή καθόλου ισχύ. Ακόμα φαίνεται και ο εικονικός διακόπτης που δίνει πλήρη ή μηδενική ένταση στην λάμπα.

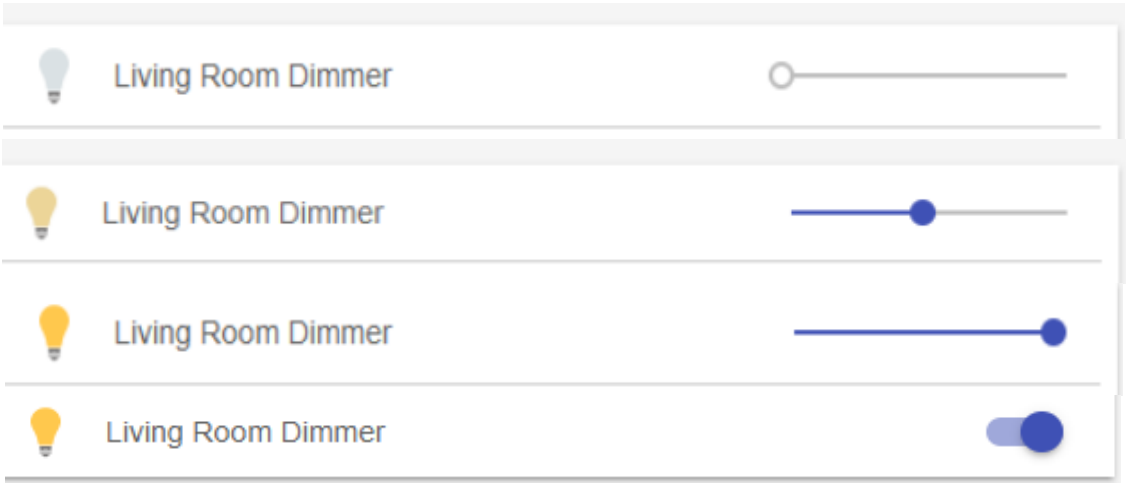

Figure 5-5 θέσεις διακοπτών

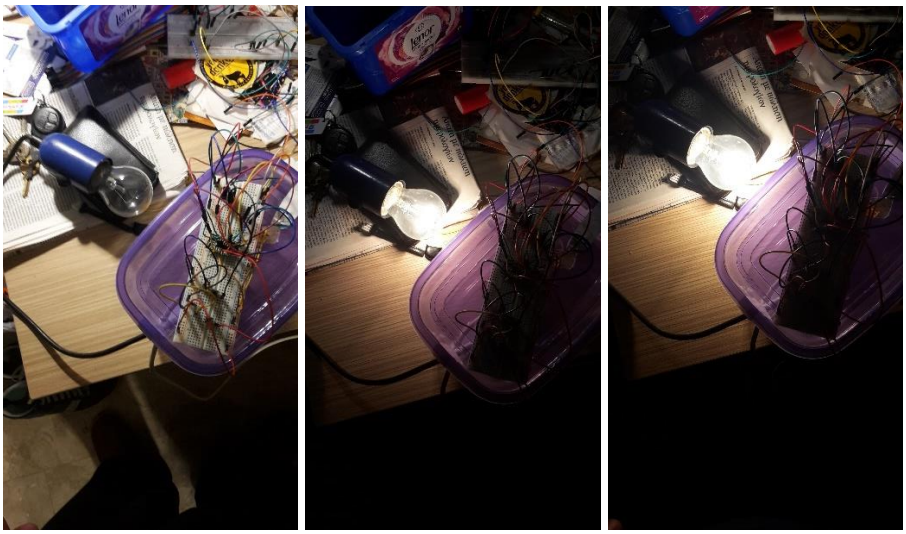

Figure 5-6 0/50/100 ένταση

5.3 Καιρικά και Αστρονομικά Δεδομένα

Μέσα στα αντίστοιχα γκρουπ της διεπαφής, βρίσκονται και τα καιρικά και αστρονομικά δεδομένα. Αυτά, πέραν του ενημερωτικού χαρακτήρα, μπορούν να χρησιμεύσουν για την ενεργοποίηση αυτοματισμών. Δηλαδή οι τιμές τους αποτελούν ερέθισμα για την αλλαγή της κατάστασης ενός αντικειμένου από την μηχανή κανόνων. Για παράδειγμα, αν τα χιλιοστά της βροχής ξεπεράσουν ένα συγκεκριμένο αριθμό, τότε να μην ενεργοποιείται το ποτιστικό σύστημα για την ημέρα εκείνη. Όλα τα δεδομένα αυτά μπορούν να βρουν μια τέτοια χρήση ανάλογα με τον χρήστη του συστήματος. Η απεικόνισή τους στην διεπαφή είναι η εξής:

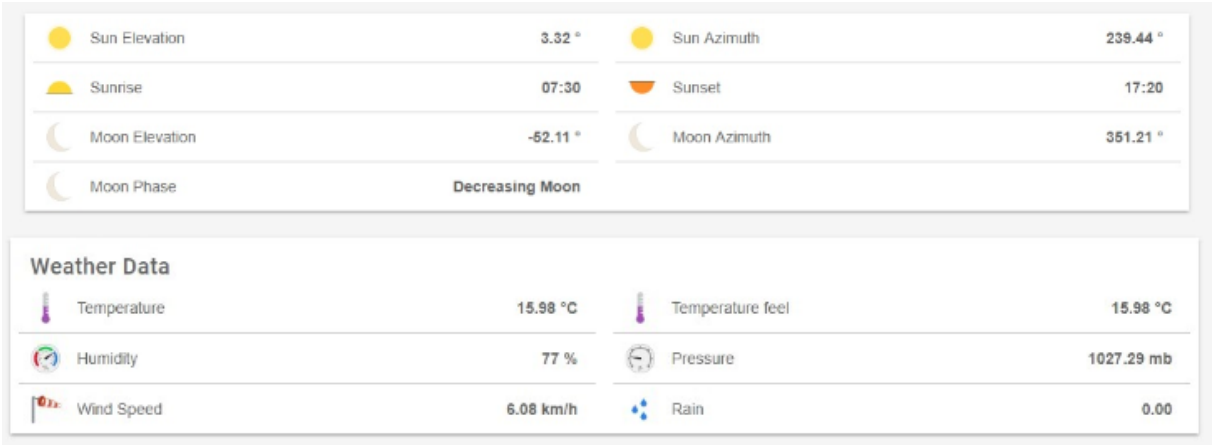

Figure 5-7 Καιρικά και Αστρονομικά δεδομένα

6 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Σε αυτό το σημείο παρουσιάζονται τα συμπεράσματα στα οποία κατέληξα κατά την διάρκεια της υλοποίησης ενός τέτοιου συστήματος οικιακού αυτοματισμού, μία αξιολόγηση του μετά την υλοποίησή του παρατηρώντας τα αποτελέσματα αλλά και επιπλέον ιδέες για την επέκτασή του με την χρήση του κατάλληλου εξοπλισμού, και μεγαλύτερης αυτοματοποίησής του.

6.1 Αξιολόγηση Υλοποίησης

Η υλοποίηση ενός τέτοιου συστήματος ήταν το δύσκολο μέρος από όλη την εργασία. Ένα σύστημα οικιακού αυτοματισμού μπορεί να είναι εξαιρετικά πολύπλευρο, καθώς δεν επικεντρώνεται σε μία τεχνολογία αλλά σε πολλές. Επίσης η διαλογή των μερών του συστήματος είναι και αυτή που καθορίζει τι τεχνολογίες θα χρησιμοποιήσει κάποιος για να το υλοποιήσει. Για να επιτευχθούν όλα όσα εφαρμόστηκαν στο σύστημα χρειάστηκαν γνώσεις σε δίκτυα, λειτουργικά συστήματα Linux, προγραμματισμού σε γλώσσα C, μικροηλεκτρονικής, αλλά και μία καλά σχηματισμένη εικόνα στο πως λειτουργούν τα Internet of Things συστήματα.

Σε κάποια σημεία της εφαρμογής ήμουνα γνωσιακά καταρτισμένος και η υλοποίηση τους ήταν εύκολη, σε άλλα όμως χρειάστηκε πολύ έρευνα στο διαδίκτυο και την παρακολούθηση εφαρμογών από άλλους, σχετικές με οικιακούς αυτοματισμούς και κατ' επέκτασιν με την δικιά μου εφαρμογή, για να κατανοήσω τα προβλήματα που προκύπταν και να ανταπεξέλθω σε αυτά.

Η τρόπος προσέγγισης ενός τέτοιου συστήματος είναι και αυτός που καθορίζει τον βαθμό δυσκολίας της ανάπτυξής του αλλά και τον τρόπο λειτουργείας. Η ανάπτυξη μιας εξατομικευμένης εφαρμογής android ή μίας διεπαφής στον ηλεκτρονικό υπολογιστή θα απαιτούσε αρκετά παραπάνω χρόνο λόγω του ότι το σύστημα είναι πολύπλευρο και απαιτεί εφαρμογές από πολλά πεδία της επιστήμης της πληροφορικής, αλλά από την άλλη αυτή η εξατομίκευση θα εμπόδιζε την ευκολία ανάπτυξης σε διαφορετικά περιβάλλοντα. Αυτό έπαιξε καθοριστικό ρόλο για την απόφαση χρήσης των διεπαφών του openHAB.

Η εφαρμογή που προέκυψε είναι μία λειτουργική εφαρμογή που υλοποιεί τρεις βασικούς παράγοντες για έναν οικιακό αυτοματισμό, δηλαδή, ένα σύστημα ασφαλείας, ένα σύστημα διαχείρισης φωτιστικών συσκευών αλλά και έντασης φωτισμού και τέλος ένα σύστημα για την διαχείριση οικιακού εξοπλισμού δύο καταστάσεων. Το τελευταίο εφαρμόστηκε με την χρήση ενός λαμπτήρα καθώς ο πειραματισμός σε συσκευές όπως το θερμοσίφωνο ή το ποτιστικό σύστημα είτε δεν ήταν ασφαλής είτε δεν υπήρχε πρόσβαση στον κατάλληλο εξοπλισμό. Παρ' όλα αυτά με ελάχιστη πλέον παρέμβαση στον κώδικα, το σύστημα μπορεί να εφαρμοστεί για την διαχείριση διαφορετικού εξοπλισμού.

Τέλος, το σύστημα αυτό με επιπλέον προσθήκες για την διαχείριση των συσκευών και με μερικές παρεμβάσεις για την βελτίωση της λειτουργείας του αλλά και της απόδοσής του, είτε από μεριάς μικροελεγκτών, είτε από μεριάς διακομιστή, μπορεί να τοποθετηθεί σε ένα σπίτι για την απομακρυσμένη διαχείρισή του εξοπλισμού του.

6.2 Μελλοντικές Επεκτάσεις

Με περισσότερο χρόνο και φυσικά εξοπλισμό το σύστημα αυτό μπορεί να περιοριστεί μόνο από την φαντασία του προγραμματιστή του. Οι επεκτάσεις για το σύστημα είναι αναγκαίες για να είναι ένα ολοκληρωμένο σύστημα οικιακού αυτοματισμού αλλά και για να είναι δυνατή η εμπορευματοποίησή του. Μερικές από τις επιπλέον εφαρμογές που θα μπορούσαν να προστεθούν είναι:

- **Servo:** με την χρήση servo motors έχουμε την δυνατότητα ακριβή ελέγχου γωνιακής ή γραμμικής τοποθέτησης, ταχύτητας και επιτάχυνσης. Με αυτά μπορούμε να ελέγξουμε τις περσίδες ενός σπιτιού για παράδειγμα. Προφανώς μπορούμε να μεγιστοποιήσουμε την αυτοματοποίηση με την εφαρμογή κανόνων στην μηχανή κανόνων του openHAB, για την αυτόματη επιλογή θέσης των περσίδων ανάλογα με την ώρα της ημέρας.
- **Voice Control:** Ένα μεγάλο πλεονέκτημα που μπορούμε να έχουμε στην εφαρμογή μας είναι η χρήση του Amazon echo dot (Alexa) για την αντικατάσταση της διεπαφής και των πραγματικών διακοπτών με φωνητικές εντολές, με μικρό σχετικά κόστος.
- **Dashboards:** Μπορούμε στο openHAB να στήσουμε την διεπαφή για dashboards τα οποία μπορούν να τοποθετηθούν διάσπαρτα μέσα σε ένα σπίτι και να αποτελούν κάτι σαν πίνακας ελέγχου των συσκευών.
- **Sonoff:** Τα sonoff μπορούνε να χρησιμοποιηθούνε για την διαχείριση ηλεκτρικών συσκευών με τρόπο παρόμοιο με αυτόν του συστήματος διαχείρισης εξοπλισμού δύο καταστάσεων με ρελέ. Αυτά θα μπορούσαν να βρουν εφαρμογή στον ηλεκτρονικό υπολογιστή και σε κάθε περιφερειακό που τον συνοδεύει, στα περιφερειακά της τηλεόρασης (ηχοσύστημα, κονσόλες) κ.α.
- **Αισθητήρες:** Ασφαλώς, η επέκταση του συστήματος απαιτεί και την προσθήκη διάφορων αισθητήρων μέσα στο σπίτι. Μία σημαντική προσθήκη είναι ο αισθητήρας καπνού που ειδοποιεί τον χρήστη σε περίπτωση φωτιάς. Επίσης η προσθήκη αισθητήρων κίνησης θα μπορούσαν να ολοκληρώσουν τον αυτοματισμό καθώς με αυτούς μπορούμε να ενεργοποιούμε το φως αν είναι νύχτα σε όποιο δωμάτιο εντοπίζεται κίνηση και να απενεργοποιείται αυτόματα όταν πλέον δεν υπάρχει κίνηση. Ακόμα μπορούν να μπουν αισθητήρες υγρασίας χώματος για την διαχείριση του ποτιστικού εξοπλισμού κ.α.

Κάθε σπίτι γενικότερα έχει διαφορετικές απαιτήσεις από ένα σύστημα διαχείρισης του εξοπλισμού του. Οι εφαρμογές που υλοποιηθήκαν με την προσθήκη μερικών από τις παραπάνω καλύπτουν το μεγαλύτερο κομμάτι των απαιτήσεων οποιουδήποτε σπιτιού για την αυτοματοποίησή του. Χρειάζεται πάντα η κατάλληλη προσαρμογή και παραμετροποίηση των μερών του συστήματος.

6.3 Συμπεράσματα

Οι εφαρμογές του Internet of Things σήμερα διαδίδονται με ραγδαίο ρυθμό και ήδη έχει ξεκινήσει η εμπορευματοποίηση των οικιακών αυτοματισμών. Στο μέλλον ίσως να βρίσκεται μέσα στα περισσότερα σπίτια. Το σύστημα που υλοποιήθηκε είναι ένα αρκετά λειτουργικό σύστημα με ελάχιστα προβλήματα στην παρούσα κατάστασή του τα οποία όμως μπορούν να ξεπεραστούν. Ο χρόνος ζωής των μικροελεγκτών είναι αρκετά καλός όσον αφορά την τροφοδοσία τους από μπαταρίες. Αυτό σε συνδυασμό την OTA ενημέρωση έχει ως αποτέλεσμα, να έχει ελαχιστοποιηθεί η ανάγκη για συντήρηση.

Η πολυπλοκότητα ενός τέτοιου πολύπλευρου συστήματος έφερε αρκετές δυσκολίες κατά την υλοποίησή του με αποτέλεσμα την απόκτηση αρκετών γνώσεων πάνω στις τεχνολογίες οι οποίες απαρτίζουν έναν οικιακό αυτοματισμό αλλά και το Internet of Things γενικότερα. Τέλος, στο μέλλον σκοπεύω στην βελτιστοποίηση της απόδοση του συστήματος αλλά και την δημιουργία frontend σε android, καθώς και την εγκατάσταση του συστήματος σε πραγματικό περιβάλλον.

Παράρτημα

Κώδικας συστήματος με ρελέ

```
/*------------------------------LIBRARIES-------------------------------------*/
#include <ESP8266WiFi.h>
#include <MQTTClient.h>
/*----------------------------------------------------------------------------*/
/*------------------------------SETTINGS--------------------------------------*/
/*WiFi Settings*/
const char* ssid = "ssid";
const char* password = "password";
/*MQTT Settings*/
char* lightsTopic = "MyHome/lights/MBR_Light";
char* rssiTopic = "MyHome/rssiLights/MBR_Light";
const char* server = "192.168.1.119";
const char* mqttDeviceID = "Device1";
/*----------------------------------------------------------------------------*/
//Relay
int relay = 5;
int relayState = LOW;
//Button
const int button = 16;
int buttonState = LOW;
int lastButtonState = HIGH;
int reading;
long lastDebounceTime = 0;
long debounceDelay = 50;
//RSSI
unsigned long lastMillis = 0;
//MQTT messages for server
char lightMsg[32];
char rssiMsg[32];
//MQTT
WiFiClient net;
MQTTClient client;
//Function to connect to WiFi & MQTT
void connect();
//Function to show the quality of the signal in percentage
int db_to_quality(int);
void setup() {
  pinMode(relay, OUTPUT);
  digitalWrite(relay, LOW);
  pinMode(button, INPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.begin(server, net);
  client.onMessage(messageReceived);
  connect();
}
void connect() {
  while(WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
  while(!client.connect(mqttDeviceID)) {
    delay(1000);
  }
  client.subscribe(lightsTopic);
}
void loop() {
  client.loop();
  delay(10);

  if(!client.connected())
    connect();
  /*--------------BUTTON-SWITCH---------------*/
  reading = digitalRead(button);
```

```
  if(reading != lastButtonState){
    lastDebounceTime = millis();
    lastButtonState = reading;
  }

  if((millis() - lastDebounceTime) > debounceDelay){
    if(buttonState != lastButtonState){
      buttonState = lastButtonState;
      if(buttonState == HIGH){
        relayState = !relayState;
        digitalWrite(relay, relayState);
        sprintf(lightMsg, "%d", relayState);
        client.publish(lightsTopic, lightMsg);
      }
    }
  }
  /*-----------------------------------------*/
  /*-------------------RSSI------------------*/
  int rssi = WiFi.RSSI();
  int quality = db_to_quality(rssi);
  sprintf(rssiMsg, "%d", quality);

  if (millis() - lastMillis > 1000) {
    lastMillis = millis();
    client.publish(rssiTopic, rssiMsg);
  }
  /*-----------------------------------------*/
}
int db_to_quality(int frssi){
  int q;
  if (frssi >= -50)
    return 4;
  else if (frssi <= -100)
    return 0;
  else if (frssi <= -50 && frssi >= -66)
    return 3;
  else if (frssi <= -67 && frssi >= -82)
    return 2;
  else if (frssi <= -83 && frssi >= -99)
    return 1;
}
void messageReceived(String &topic, String &payload) {
  String msg = payload;
  if (msg == "L1ON") {
    digitalWrite(relay, HIGH);
    relayState = !relayState;
    delay(250);
  }
  else if(msg == "L1OFF") {
    digitalWrite(relay, LOW);
    relayState = !relayState;
    delay(250);
  }
}
```

Κώδικας Συστήματος Ασφαλείας

```
/*-----------------------------LIBRARIES---------------------------------------*/

#include <ESP8266WiFi.h>
#include <MQTTClient.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
/*-------------------------------------------------------------------------------*/
/*-----------------------------SETTINGS----------------------------------------*/
/*WiFi Settings*/
const char* ssid = "ssid";
const char* password = "password";
/*MQTT Settings*/
char* sensorTopic = "MyHome/doors/Main_Door";
char* securityTopic = "MyHome/security/Main_Door";
char* rssiTopic = "MyHome/rssiDoors/Main_Door";
const char* server = "192.168.1.107";
const char* mqttDeviceID = "Device2";
/*-------------------------------------------------------------------------------*/
int gpio0_pin = 0;
int gpio2_pin = 2;
//Magnetic Reed Switch
int sensor = 12;
boolean opened = true;
boolean closed;
//LED
int led = 4;
boolean secFlag = false;
//RSSI
unsigned long lastMillis = 0;
//OTA
bool ota_flag = true;
int time_elapsed = 0;
//MQTT messages for server
char rssiMsg[32];
//MQTT
WiFiClient net;
MQTTClient client;
//Function to connect to WiFi & MQTT
void connect();
//Function to show the quality of the signal in percentage
int db_to_quality(int);
void setup() {
  // preparing GPIOs
  pinMode(gpio0_pin, OUTPUT);
  digitalWrite(gpio0_pin, HIGH);
  pinMode(gpio2_pin, OUTPUT);
  digitalWrite(gpio2_pin, HIGH);

  pinMode(sensor, INPUT);
  digitalWrite(sensor, LOW);
  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.begin(server, net);
  client.onMessage(messageReceived);
  ArduinoOTA.setPassword((const char *)"password2");
  connect();
  ArduinoOTA.onStart([]() {
    digitalWrite(led, HIGH);
  });
  ArduinoOTA.onEnd([]() {
    digitalWrite(led, LOW);
  });
  ArduinoOTA.onError([](ota_error_t error) {
```

```
    ESP.restart();
  });
  ArduinoOTA.begin();
}
void connect() {
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
  while (!client.connect(mqttDeviceID)) {
    delay(1000);
  }
  client.subscribe(securityTopic);
}
void loop() {
  if (ota_flag) {
    while (time_elapsed < 15000) {
      ArduinoOTA.handle();
      time_elapsed = millis();
      delay(10);
    }
    ota_flag = false;
  }
  client.loop();
  delay(10);
  if(secFlag){
    digitalWrite(led, HIGH);
    delay(200);
    digitalWrite(led, LOW);
    delay(200);
  }
  if (!client.connected())
    connect();
  if (digitalRead(sensor) == HIGH) {
    if (opened) {
      opened = false;
      client.publish(sensorTopic, "OPEN");
      delay(50);
    }
    closed = true;
  }
  if (digitalRead(sensor) == LOW) {
    if (closed) {
      closed = false;
      client.publish(sensorTopic, "CLOSED");
      delay(50);
    }
    opened = true;
  }
  /*-------------------RSSI--------------------*/
  int rssi = WiFi.RSSI();
  int quality = db_to_quality(rssi);
  //from int to char
  sprintf(rssiMsg, "%d", quality);
  if (millis() - lastMillis > 1000) {
    lastMillis = millis();
    client.publish(rssiTopic, rssiMsg);
  }
  /*-------------------------------------------*/
}
int db_to_quality(int frssi) {
  int q;
  if (frssi >= -50)
    return 4;
  else if (frssi <= -100)
    return 0;
  else if (frssi <= -50 && frssi >= -66)
    return 3;
  else if (frssi <= -67 && frssi >= -82)
```

```
    return 2;
  else if (frssi <= -83 && frssi >= -99)
    return 1;
}
void messageReceived(String &topic, String &payload) {
  String msg = payload;
  if (msg == "secON") {
    digitalWrite(led, HIGH);
    secFlag = true;
    delay(2000);
  }
  else if (msg == "secOFF") {
    digitalWrite(led, LOW);
    secFlag = false;
    delay(250);
  }
}
```

Κώδικας Συστήματος Ελέγχου Φωτιστικών με Dimmer

```
/*-----------------------------LIBRARIES---------------------------------------*/
#include <ESP8266WiFi.h>
#include <MQTTClient.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include "OneButton.h"
/*-----------------------------------------------------------------------------*/
/*-----------------------------SETTINGS----------------------------------------*/
/*WiFi Settings*/
const char* ssid = "ssid";
const char* password = "password";
/*MQTT Settings*/
char* lightsTopic = "MyHome/lights/LVR_Light";
char* dimButtonsTopic = "MyHome/lights/DMB_Light";
char* rssiTopic = "MyHome/rssiLights/LVR_Light";
const char* server = "192.168.1.119";
const char* mqttDeviceID = "Device3";

/*-----------------------------------------------------------------------------*/

int gpio0_pin = 0;
int gpio2_pin = 2;

//Light Circuit
int AC_pin = 12;
int zc = 13;
byte dim = 0;
//OTA
bool ota_flag = true;
int time_elapsed = 0;
//Button
OneButton increase(16, false);
OneButton decrease(4, false);
int maxInc;
int minDec;
//RSSI
unsigned long lastMillis = 0;   //Millisec counter
//MQTT messages for server
char lightMsg[32];
char rssiMsg[32];
//MQTT
WiFiClient net;
MQTTClient client;
//Function to connect to WiFi & MQTT
void connect();
//Function to show the quality of the signal in percentage
int db_to_quality(int);
void setup() {
  // preparing GPIOs
  pinMode(gpio0_pin, OUTPUT);
  digitalWrite(gpio0_pin, HIGH);
  pinMode(gpio2_pin, OUTPUT);
  digitalWrite(gpio2_pin, HIGH);
  Serial.begin(9600);
  pinMode(AC_pin, OUTPUT);
  pinMode(zc, INPUT_PULLUP);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.begin(server, net);
  client.onMessage(messageReceived);
  ArduinoOTA.setPassword((const char *)"password2");
  connect();
  ArduinoOTA.onStart([]() {
  });
  ArduinoOTA.onEnd([]() {
```

```
  });
  ArduinoOTA.onError([](ota_error_t error) {
    ESP.restart();
  });
  ArduinoOTA.begin();
}
void light() {
  if (dim < 1) {
    digitalWrite(AC_pin, LOW);      //Turn TRIAC completely OFF if dim is 0
  }
  if (dim > 254) {
    digitalWrite(AC_pin, HIGH);     //Turn TRIAC completely ON if dim is 255
  }
  if (dim > 0 && dim < 255) {
    delayMicroseconds(37 * (255 - dim));    //Dimming part, if dim is not 0 and not
255
    digitalWrite(AC_pin, HIGH);
    delayMicroseconds(15);
    digitalWrite(AC_pin, LOW);
  }
}
void clickA() {
  float x;
  int percentage;
  if (dim <= 245) {
    dim += 10;
  }
  else if (dim > 250) {
    dim = 255;
  }
  x = (dim * 100) / 255;
  percentage = int(x);
  sprintf(lightMsg, "%d", percentage);
  client.publish(dimButtonsTopic, lightMsg);
}
void clickB() {
  float x;
  int percentage;
  if (dim >= 10) {
    dim -= 10;
  }
  else if (dim < 10) {
    dim = 0;
  }
  x = (dim * 100) / 255;
  percentage = int(x);
  sprintf(lightMsg, "%d", percentage);
  client.publish(dimButtonsTopic, lightMsg);
}
void pressA() {
  dim = 255;
  int x = 100;
  delay(5);
  sprintf(lightMsg, "%d", x);
  client.publish(dimButtonsTopic, lightMsg);
}
void pressB() {
  dim = 0;
  int x = 0;
  delay(5);
  sprintf(lightMsg, "%d", x);
  client.publish(dimButtonsTopic, lightMsg);
}
void doubleClick() {
  dim = 128;
  int x = 50;
  delay(5);
  sprintf(lightMsg, "%d", x);
```

```
  client.publish(dimButtonsTopic, lightMsg);
}
void connect() {
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
  while (!client.connect(mqttDeviceID)) {
    delay(1000);
  }
  client.subscribe(lightsTopic);
}
void loop() {
  if (ota_flag) {
    while (time_elapsed < 15000) {
      ArduinoOTA.handle();
      time_elapsed = millis();
      delay(10);
    }
    ota_flag = false;
    attachInterrupt(zc, light, FALLING);
    increase.attachClick(clickA);
    increase.attachPress(pressA);
    increase.attachDoubleClick(doubleClick);
    decrease.attachClick(clickB);
    decrease.attachPress(pressB);
    decrease.attachDoubleClick(doubleClick);
  }
  if (!client.connected())
    connect();
  client.loop();
  /*-----------------BUTTONS------------------*/
  increase.tick();
  decrease.tick();
  /*------------------------------------------*/
  /*-------------------RSSI-------------------*/
  int rssi = WiFi.RSSI();
  int quality = db_to_quality(rssi);
  //from int to char
  sprintf(rssiMsg, "%d", quality);
  if (millis() - lastMillis > 1000) {
    lastMillis = millis();
    client.publish(rssiTopic, rssiMsg);
  }
  /*------------------------------------------*/
}
int db_to_quality(int frssi) {
  int q;
  if (frssi >= -50)
    return 4;
  else if (frssi <= -100)
    return 0;
  else if (frssi <= -50 && frssi >= -66)
    return 3;
  else if (frssi <= -67 && frssi >= -82)
    return 2;
  else if (frssi <= -83 && frssi >= -99)
    return 1;
}
void messageReceived(String &topic, String &payload) {
  String msg = payload;
  int fdim;
  float multiplier, percentage;
  percentage = msg.toInt();
  multiplier = percentage / 100;
  fdim = (int)255 * multiplier;
  dim = (byte)fdim;
}
```

Items File

```
//Groups
Group gFF               "First Floor"           <firstfloor>
Group gGF               "Ground Floor"          <groundfloor>
Group gLiving           "Living Room"           <sofa>          (gFF)
Group gMyroom           "Adam's Room"           <bedroom>       (gGF)
Group gWeather          "Weather"               <sun_clouds>
Group gAstronomy        "Astronomical Data"     <sun>
Group gLights           "House Lights"          <light>
Group gRSSI             "Items RSSI"            <signal>
Group gMyroom_RSSI      "My Room Items RSSI"    <signal>        (gRSSI, gMyroom)
Group gLiving_RSSI      "Living Room Iems RSSI" <signal>        (gRSSI, gLiving)

//Active Groups
Group:Switch:OR(ON, OFF)         Lights  "All Lights [(%d)]"
Group:Contact:OR(OPEN, CLOSED)   Doors   "Open Doors [(%d)]"    <door>
Group:Contact:OR(OPEN, CLOSED)   Windows "Open Windows [(%d)]"  <contact>

//Lights
Switch MBR_light    "Bedroom Ceiling Light"         <light>         (gMyroom, Lights, gLights)
{mqtt=">[broker:MyHome/lights/MBR_Light:command:ON:L1ON],>[broker:MyHome/lights/MBR_Light:comm
and:OFF:L1OFF],<[broker:MyHome$

Dimmer LVR_light                "Living Room Dimmer"            <slider>        (gLiving)
{mqtt=">[broker:MyHome/lights/LVR_Light:state:*:default],<[broker:MyHome/lights/DMB_Light:stat
e:default]"}
Switch Dummy1      "Living Room Dimmer"           <light>         (gLights, Lights)

//Security<>
Switch    SecuritySys                           "Security"                              <siren>
{mqtt=">[broker:MyHome/security/Main_Door:command:ON:secON],>[broker:MyHome/security/Main_Door
:command:OFF:secOFF]"}
Contact MainDoorSensor            "Main Door [%s]"          <frontdoor>      (Doors, gLiving)
{mqtt="<[broker:MyHome/doors/Main_Door:state:default]"}

//Weather<>
Number Temperature              "Temperature [%.2f °C]"         <temperature>   (gWeather)
{weather="locationId=home, type=temperature, property=current"}
Number Temperature_Feel "Temperature feel [%.2f °C]"    <temperature>   (gWeather)
{weather="locationId=home, type=temperature, property=feel"}
Number Humidity     "Humidity [%d %%]"                  <humidity>      (gWeather)
{weather="locationId=home, type=atmosphere, property=humidity"}
Number Pressure       "Pressure [%.2f mb]"              <pressure>      (gWeather)
{weather="locationId=home, type=atmosphere, property=pressure"}
Number Wind            "Wind Speed [%.2f km/h]"         <wind>          (gWeather)
{weather="locationId=home, type=wind, property=speed"}
Number Rain            "Rain [%.2f] mm/h"               <rain>          (gWeather)
{weather="locationId=home, type=precipitation, property=rain"}

//Astronomical Data
Number   Sun_Elevation    "Sun Elevation"     <sun>          (gAstronomy)          {
channel = "astro:sun:home:position#elevation" }
Number   Sun_Azimuth            "Sun Azimuth"      <sun>          (gAstronomy)     {
channel = "astro:sun:home:position#azimuth" }
DateTime Sunrise_Time           "Sunrise [%1$tH:%1$tM]"     <sunrise>  (gAstronomy)
{ channel = "astro:sun:home:rise#start" }
DateTime Sunset_Time            "Sunset [%1$tH:%1$tM]"   <sunset> (gAstronomy)     {
channel = "astro:sun:home:set#start" }
Number   Moon_Elevation         "Moon Elevation"   <moon>         (gAstronomy)     {
channel = "astro:moon:home:position#elevation" }
Number   Moon_Azimuth           "Moon Azimuth"     <moon>         (gAstronomy)     {
channel = "astro:moon:home:position#azimuth" }
String   Moon_Phase    "Moon Phase [MAP(astro.map):%s]" <moon>          (gAstronomy)
{ channel = "astro:moon:home:phase#name" }

//Date & Time
DateTime CurrentDate    "Date [%1$tA, %1$td.%1$tm.%1$tY, %1$tR]"   <calendar>      {
channel="ntp:ntp:home:dateTime" }
```

```
//WiFi Appliances
Number MBR_BCL_RSSI      "Ceiling Light [%d/4]"    <signal>          (gMyroom_RSSI)
{mqtt="<[broker:MyHome/rssiLights/MBR_Light:state:default]"}

String RSSI_Current_MBR_BCL  "[MAP(rssi.map):%s]"    <number>         (gMyroom_RSSI)
{mqtt="<[broker:MyHome/rssiLights/MBR_Light:state:default]"}

Number LR_HMD_RSSI   "Main door [%d/4]"              <signal>         (gLiving_RSSI)
{mqtt="<[broker:MyHome/rssiDoors/Main_Door:state:default]"}

String RSSI_Current    "[MAP(rssi.map):%s]"          <number>         (gLiving_RSSI)
{mqtt="<[broker:MyHome/rssiDoors/Main_Door:state:default]"}

Number LR_LCL_RSSI               "Dimmer [%d/4]"          <signal>    (gLiving_RSSI)
{mqtt="<[broker:MyHome/rssiLights/LVR_Light:state:default]"}
```

Βιβλιογραφία

1) http://homeautomationforgeeks.com/index.php
2) https://github.com/openhab/openhab1-addons/wiki/Explanation-of-items#real-life-example
3) https://community.openhab.org/t/slider-with-dimmer/6424/8
4) https://docs.openhab.org/
5) http://esp8266.github.io/Arduino/versions/2.3.0/doc/filesystem.html
6) https://www.olimex.com/Products/IoT/ESP8266-EVB/resources/ESP8266-EVB-how-to-use-Arduino.pdf
7) https://pubsubclient.knolleary.net/api.html
8) https://www.raspberrypi.org/blog/pi-3-booting-part-i-usb-mass-storage-boot/
9) https://github.com/openhab/openhab1-addons/wiki/Actions#mqtt-action
10) http://www.esp8266.com/wiki/doku.php?id=esp8266_gpio_pin_allocations
11) http://www.electrodragon.com/w/ESP-12F_ESP8266_Wifi_Board
12) https://github.com/openhab/openhab-distro/tree/master/features/distro-resources/src/main/resources
13) http://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/
14) http://whatis.techtarget.com/definition/debouncing
15) https://myelectronicslab.com/esp8266-ota-example-arduino-code-tutorial-air-update/
16) https://github.com/esp8266/Arduino/issues/1017
17) http://www.mathertel.de/Arduino/OneButtonLibrary.aspx
18) https://www.sas.com/el_gr/insights/big-data/internet-of-things.html
19) https://en.wikipedia.org/wiki/Internet_of_things
20) https://www.quora.com/What-is-the-difference-between-WSN-and-IOT
21) https://www.linkedin.com/pulse/internet-things-iot-characteristics-kavyashree-g-c/
22) https://en.wikipedia.org/wiki/Wireless_sensor_network
23) https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/
24) http://www.electronicdesign.com/iot/understanding-protocols-behind-internet-things
25) https://www.postscapes.com/internet-of-things-protocols/
26) https://www.element14.com/community/groups/internet-of-things/blog/2017/02/05/iot-protocols-an-overview
27) https://en.wikipedia.org/wiki/MQTT
28) https://en.wikipedia.org/wiki/Constrained_Application_Protocol
29) https://www.standardsuniversity.org/e-magazine/march-2016/internet-of-things-requirements-and-protocols/
30) http://ess-wiki.advantech.com.tw/view/SW_Service/Azure/IoT_Suite
31) https://www.elecrow.com/download/ESP-12F.pdf
32) https://en.wikipedia.org/wiki/Servomotor
33) Σχεδίαση και ανάπτυξη πλατφόρμας Ασύρματου Δικτύου Αισθητήρων με χρήση expander(MDA300) για συλλογή και επεξεργασία Αναλογικών και Ψηφιακών σημάτων, Παπαδογιάννης Νικόλαος, Κοντάκης Ελευθερία
34) Εφαρμογή Οικιακού Αυτοματισμού, Κορδαλής Παρασκευάς, Μπιλίκμπασι Ρενάτο
35) Raspberry Pi Home Automation with Arduino, Andrew k. Dennis
36) OpenHAB Home Automation on Raspberry Pi, James Bruce
37) OpenHAB Beginner's Guide part 1, James Bruce
38) OpenHAB Beginner's Guide part 2, James Bruce
39) Internet of Things – IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges, Keyur K Patel, Sunil M Patel
40) A guide to Debouncing, Jack G Ganssle