ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σύστημα οικιακού αυτοματισμού (home automation) βασιζόμενο στη πλατφόρμα Raspberry Pi

Διπλωματική Εργασία

Αναστασιάδης Παναγιώτης

Επιβλέπων: Θάνος Γεώργιος

Βόλος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

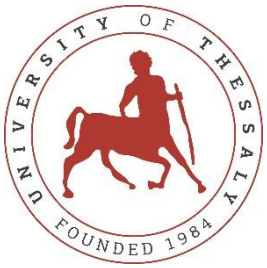
Σύστημα οικιακού αυτοματισμού (home automation) βασιζόμενο στη πλατφόρμα Raspberry Pi

Διπλωματική Εργασία

Αναστασιάδης Παναγιώτης

Επιβλέπων: Θάνος Γεώργιος

Βόλος 2021



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A home automation system based on Raspberry Pi

Diploma Thesis

Anastasiadis Panagiotis

Supervisor: Thanos Georgios

Volos 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Θάνος Γεώργιος** Μέλος ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Τουσίδου Ελένη** Μέλος ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Φεύγας Αθανάσιος** Μέλος ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 25-02-2021

ΕΥΧΑΡΙΣΤΙΕΣ

Θέλω να ευχαριστήσω θερμά την οικογένεια μου για την στήριξη, την αγάπη και την πολύτιμη βοήθεια τους, οι οποίοι είναι πάντα εκεί για μένα, στις σπουδές μου, στα όνειρα μου και στη ζωή μου.

Θα ήθελα επίσης να ευχαριστήσω τον επιβλέποντα καθηγητή μου Θάνο Γεώργιο για τη συνέπεια του και την άριστη συνεργασία μας, κατά την περάτωση της διπλωματικής μου εργασίας.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

Αναστασιάδης Παναγιώτης

Φεβρουάριος 2021

ΠΕΡΙΛΗΨΗ

Το Διαδίκτυο των Πραγμάτων αφορά ένα σύνολο τεχνολογιών που επιτρέπει τη διασύνδεση διαφόρων συσκευών στο Διαδίκτυο, των οποίων η λειτουργία σχετίζεται κυρίως με τη συλλογή δεδομένων από το φυσικό περιβάλλον, την μεταφορά αυτών πάνω από το δίκτυο, και την μετέπειτα επεξεργασία, παρακολούθηση και ανάλυση τους. Ο τομέας των οικιακών αυτοματισμών είναι μια από τις πολλές εφαρμογές που βρίσκονται κάτω από την «ομπρέλα» του Διαδικτύου των Πραγμάτων και βασίζεται στην ιδέα της κατασκευής «έξυπνων» σπιτιών που στόχο έχουν την παροχή διευκολύνσεων και καλύτερου ελέγχου στους διαμένοντες σε αυτά. Στόχος της παρούσας διπλωματικής εργασίας είναι να προσθέσει στις ήδη διαθέσιμες λύσεις και υπηρεσίες που υπάρχουν στο πλαίσιο των οικιακών αυτοματισμών, ένα σύστημα το οποίο υλοποιείται από το μηδέν, βασίζεται σε hardware χαμηλού κόστους και προσφέρει από τη μια πλευρά ευελιξία μέσω των επιλογών σε αυτοματισμούς που παρέχει στον χρήστη, και από την άλλη, τον απόλυτο έλεγχο γύρω από τη δομή και τη λειτουργία του. Η φύση του συστήματος βασίζεται στην ύπαρξη ενός κεντρικού server, την αρμοδιότητα του οποίου αναλαμβάνει ο υπολογιστής Raspberry Pi, καθώς και στην ασύρματη επικοινωνία αυτού με τους διάφορους κόμβους ή sensor nodes που βρίσκονται κατανεμημένοι στο σπίτι και απαρτίζονται από microcontroller boards και αισθητήρες συνδεμένους πάνω σε αυτά. Τέλος, για τον έλεγχο του συστήματος, κατασκευάζεται ένας web client και μια native Android εφαρμογή που δίνουν στον χρήστη την ευχέρεια να διαχειρίζεται το σύστημα και τους οικιακούς αυτοματισμούς που θέτει, μέσα από τις καθημερινές συσκευές του, όπως laptops, smartphones, tablets κ.ά.

ABSTRACT

The Internet of Things is an aggregation set of technologies that allows the interconnection of various devices over the Internet, the operation of which is mainly related to the collection of data coming from the natural environment. These data are then transferred over the network for their subsequent processing, monitoring and analysis. Home automation is one of the many applications that lives under the technological field of Internet of Things and is based upon the idea of building Smart Homes that aim to provide convenience and better control to their residents. The main purpose of the Thesis is to add next to the already available home automation services out there, a system that is being implemented from scratch, depends on low-cost hardware, offers flexibility through its handy automation options and provides full control over its structure and operations. The system's nature includes a base server running on Raspberry Pi computer that is responsible for the wireless communication with the various sensor nodes located inside the house. These sensor nodes consist of microcontroller boards and several sensors connected to them. Lastly, both a web client and a native Android application are developed, in order to provide the user with the ability of managing the system and setting automation actions. These client applications are designed to run on everyday devices such as smartphones, laptops, tablets, etc.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Το Διαδίκτυο των Πραγμάτων ή στα αγγλικά Internet of Things (IoT) είναι ένας τεχνολογικός κλάδος που έχει γνωρίσει ευρεία ανάπτυξη τα τελευταία χρόνια. Ο πυρήνας του βασίζεται στη λογική ύπαρξης ενός περιβάλλοντος στο οποίο διαφορετικές συσκευές, αντλώντας πληροφορίες από το φυσικό περιβάλλον, μπορούν να διασυνδέονται και να επικοινωνούν πάνω από ένα δίκτυο ή το διαδίκτυο, ανταλλάσσοντας δεδομένα. Ήδη σε όλον τον κόσμο βρίσκονται πλέον δισεκατομμύρια συσκευές διασυνδεμένες κάτω από κάποιο δίκτυο και αυτές συναντώνται σε μια τεράστια γκάμα εφαρμογών που αφορούν την υγειονομική περίθαλψη, τα «έξυπνα» σπίτια, την περιβαλλοντική παρακολούθηση, τις μεταφορές, τη βιομηχανία, κ.ά. [1].

Ο τομέας των «έξυπνων σπιτιών» που αποτελεί μια από τις εφαρμογές που ανήκουν στο Διαδίκτυο των Πραγμάτων, στηρίζεται στην υλοποίηση συστημάτων οικιακών αυτοματισμών που έχουν ως στόχο την παροχή ανέσεων, διευκολύνσεων και κατά επέκταση ποιοτικότερου βιοτικού επιπέδου στους ανθρώπους που τα χρησιμοποιούν, όσον αφορά το περιβάλλον της οικίας που διαμένουν και της αλληλεπίδρασης του με αυτό. Οι οικιακοί αυτοματισμοί συνήθως σχεδιάζονται γύρω από τον έλεγχο συσκευών ή αισθητήρων όπως φωτιστικά, θερμοστάτες, αισθητήρες θερμοκρασίας-υγρασίας, αισθητήρες κίνησης και άλλες ηλεκτρικές συσκευές που συναντώνται σε ένα σπίτι. Επιπλέον, ο έλεγχος αυτός τείνει να γίνεται ασύρματα και μέσα από τη χρήση καθημερινών συσκευών της σύγχρονης εποχής, όπως smartphones, tablets, laptops ή desktop υπολογιστές [2].

Δημοφιλής εταιρίες όπως η Amazon, η Google, η Samsung και η Apple προσφέρουν έτοιμα συστήματα οικιακών αυτοματισμών διαθέσιμα προς αγορά, ενώ υφίστανται και open-source πλατφόρμες που σχετίζονται με το home automation, όπως το openHab, το Home Assistant και το Domoticz. Οι πλατφόρμες αυτές υποστηρίζουν μεγάλο εύρος συσκευών, αλλά συνήθως απαιτούν προεργασία για τη σωστή ρύθμιση και εγκατάσταση τους.

Σκοπός της παρούσας διπλωματικής εργασίας είναι να συνεισφέρει στο τομέα των οικιακών αυτοματισμών, δείχνοντας ότι είναι εφικτή η υλοποίηση ενός συστήματος που βασίζεται εξ ολοκλήρου σε φθηνό hardware και ταυτόχρονα προσφέρει σημαντικές

διευκολύνσεις και αυτοματισμούς στον χρήστη που το χρησιμοποιεί. Ακόμη, το σύστημα δεν περιορίζεται στις υπάρχουσες open-source home automation πλατφόρμες και κατασκευάζεται εξ ολοκλήρου από την αρχή, δίνοντας έτσι στον χρήστη τον απόλυτο έλεγχο ως προς αυτό και τις λειτουργίες του.

Έτσι λοιπόν, βάσει του περιεχομένου που πραγματεύεται, τα υπόλοιπα κεφάλαια της διπλωματικής εργασίας διαρθρώνονται ως εξής:

Στο Κεφάλαιο 2 προσδιορίζεται η δομή του συστήματος, περιγράφονται οι απαιτήσεις που είναι επιθυμητό να καλύπτει και γίνεται αναφορά στη μεθοδολογία που ακολουθείται κατά την υλοποίηση.

Το Κεφάλαιο 3 εστιάζει στην επιλογή του hardware, την ρύθμιση και το στήσιμο του στο σπίτι και στο Κεφάλαιο 4 προσδιορίζεται το σύνολο των software εργαλείων που θα χρησιμοποιηθούν για να κατασκευάσουν τον κεντρικό server του συστήματος. Στο Κεφάλαιο 4, παρουσιάζεται ακόμη η αρχικοποίηση της server εφαρμογής και η σύνδεση του συστήματος με μια βάση δεδομένων.

Τα Κεφάλαια 5 και 6 περιστρέφονται γύρω από το MQTT πρωτόκολλο επικοινωνίας. Στο Κεφάλαιο 5 εξηγείται αναλυτικά το τί ορίζεται ως MQTT και ποιες είναι οι λειτουργίες του, ενώ στο Κεφάλαιο 6 αναλύεται η σημασία του στη παρούσα εφαρμογή και πώς χρησιμοποιείται για την επικοινωνία των συσκευών με τον κεντρικό server του συστήματος.

Στα Κεφάλαια 7 και 8 παρουσιάζεται η υλοποίηση του web client του συστήματος και οι τρόποι που τον χρησιμοποιεί ο χρήστης για την παρακολούθηση των αισθητήρων, αλλά και τον ορισμό αυτοματισμών γύρω από αυτούς.

Το Κεφάλαιο 9 αφορά την υλοποίηση μιας native Android εφαρμογής για το σύστημα, η οποία διαθέτει όλες τις δυνατότητες του web client και ταυτόχρονα προστίθεται σε αυτή ένα ακόμα είδους αυτοματισμού που βασίζεται στην τοποθεσία του χρήστη.

Το Κεφάλαιο 10 αναλύει τον τρόπο με τον οποίον τόσο ο web client όσο και η Android εφαρμογή επικοινωνούν με τον κεντρικό server, διαδικασία που ουσιαστικά αποτελεί τον βασικό πυρήνα της λειτουργίας τους.

Τα Κεφάλαιο 11 και 12 αφορούν την ολοκλήρωση της ανάπτυξης του συστήματος. Από τη μία πλευρά, το Κεφάλαιο 11 εστιάζει στην έκθεση του συστήματος στο διαδίκτυο, ώστε ο

χρήστης να έχει πρόσβαση σε αυτό από οπουδήποτε, ενώ το Κεφάλαιο 12 περιγράφει πρακτικά σενάρια για το ολοκληρωμένο πλέον σύστημα οικιακών αυτοματισμών.

Τέλος, το Κεφάλαιο 13 αφορά τα συμπεράσματα της διπλωματικής εργασίας σχετικά με το τελικό αποτέλεσμα και πώς αυτό ανταποκρίνεται στους στόχους που τέθηκαν. Παράλληλα αναφέρονται μερικές από τις δυσκολίες κατά την ανάπτυξη, όπως επίσης και πιθανές προτάσεις για περαιτέρω λειτουργικές προσθήκες στο σύστημα μελλοντικά.

ΚΕΦΑΛΑΙΟ 2

ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΟΙΚΙΑΚΩΝ ΑΥΤΟΜΑΤΙΣΜΩΝ

2.1 Λειτουργίες και δυνατότητες του συστήματος

Όπως υποδεικνύει και το όνομα του, ένα σύστημα οικιακών αυτοματισμών βασίζεται στην ικανότητα του να παρέχει στον χρήστη εύχρηστους αυτοματισμούς που διευκολύνουν την αλληλεπίδραση του με το σπίτι του. Αυτό δύναται να επιτευχθεί, εφόσον είναι εγκατεστημένη η σχετική υποδομή στο σπίτι και ο χρήστης μέσω μιας εφαρμογής που μπορεί να χρησιμοποιεί μέσα από τις ηλεκτρονικές συσκευές του (κινητό, tablet, laptop, σταθερός υπολογιστής κ.α.), είναι δυνατόν να ελέγξει τις οικιακές συσκευές του ή να προγραμματίσει ενέργειες που εκτελούνται αυτόματα.

Μέσα από την παρούσα διπλωματική εργασία, επιχειρείται να κατασκευαστεί εξ αρχής ένα τέτοιο σύστημα, το οποίο θα μπορεί ο χρήστης να το εγκαταστήσει στην οικεία του και θα έχει πρόσβαση σε αυτό μέσω μιας web εφαρμογής, διαθέσιμης σε όλες τις συσκευές του που διαθέτουν web browser, ή εναλλακτικά μέσα από μια εφαρμογή εγκατεστημένη στις Android συσκευές του. Με την ολοκλήρωση του συστήματος που παρουσιάζεται, ο χρήστης θα πρέπει να έχει τις ακόλουθες δυνατότητες:

- Παρακολούθηση μετρήσεων, σχετικών με τη κατάσταση του σπιτιού (θερμοκρασία, υγρασία, φωτεινότητα, ανίχνευση κίνησης κλπ.).
- Εκτέλεση ενεργειών μέσα από το γραφικό περιβάλλον της εφαρμογής στις υπάρχουσες συσκευές του (π.χ. άνοιγμα/κλείσιμο λάμπας).
- Προγραμματισμός ενεργειών με βάση τον χρόνο (π.χ. άνοιγμα λάμπας στις 8 μ.μ.).
- Προγραμματισμός ενεργειών με βάση τις τιμές άλλων αισθητήρων (π.χ. άνοιγμα της θέρμανσης όταν η θερμοκρασία πέσει κάτω από 20 βαθμούς Κελσίου).
- Προγραμματισμός ενεργειών με βάση την τοποθεσία. Η λειτουργία αυτή αφορά την εφαρμογή του κινητού τηλεφώνου. Ο χρήστης μπορεί να ορίσει πως όταν βρίσκεται σε μια συγκεκριμένη ακτίνα γύρω από τη τοποθεσία της οικίας του, να ενεργοποιείται η εκτέλεση μιας ενέργειας (π.χ. άνοιγμα θέρμανσης όταν βρίσκεται σε ακτίνα 300 μέτρων από το σπίτι του).

- Πρόσβαση όλων των παραπάνω, τόσο από το προσωπικό του δίκτυο όσο και από οποιοδήποτε άλλο εξωτερικό δίκτυο για όταν αυτός βρίσκεται εκτός της οικείας του.

2.2 Απαιτήσεις και χαρακτηριστικά του συστήματος

Κατά την κατασκευή του συστήματος, συγκεκριμένες παράμετροι πρέπει να ληφθούν υπόψη. Αυτές οι παράμετροι αφορούν λεπτομέρειες για τη φύση του, όπως το κόστος, ο χώρος που θα καταλαμβάνει και οι δυνατότητες επέκτασης του. Έτσι, το σύστημα μας επιχειρεί να καλύψει τις εξής απαιτήσεις:

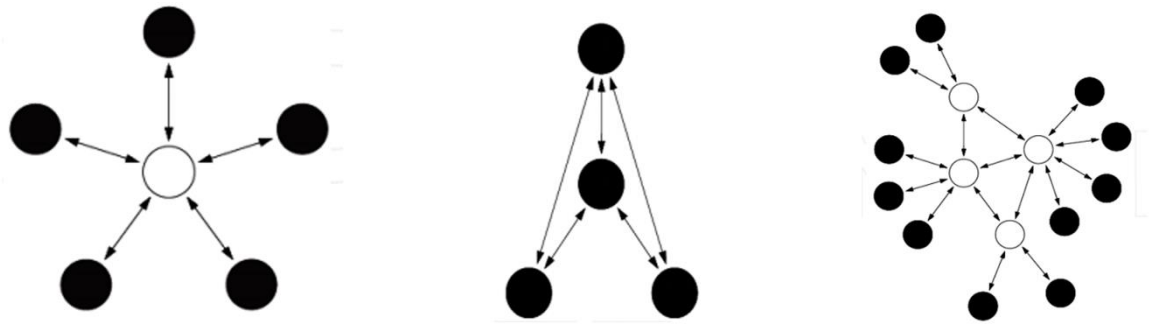
- Κρατάει το κόστος σε χαμηλά επίπεδα. Αυτό συμβαίνει γιατί θα βασίζεται σε φθηνό hardware, το οποίο μπορεί να αγοραστεί και να αντικατασταθεί εύκολα εξαιτίας αυτής του της ιδιότητας, αλλά ταυτόχρονα θα παραμένει αξιόπιστο και εύχρηστο ως προς τις ανάγκες της εφαρμογής. Επιπρόσθετα, ο τομέας του software τόσο για το frontend (η υλοποίηση client εφαρμογής για web browsers και android κινητά τηλέφωνα) όσο και για το backend επίπεδο (η κατασκευή του κεντρικού server) του συστήματος θα βασίζεται είτε σε open-source βιβλιοθήκες κώδικα είτε σε δωρεάν υπηρεσίες που προσφέρουν closed-source εργαλεία.
- Στηρίζεται στην ασύρματη επικοινωνία μεταξύ του κεντρικού server και των συσκευών. Επομένως, εξαλείφει τυχόν περιορισμούς που αφορούν το χώρο της οικίας που εγκαθίσταται και ως ένα ελεύθερο από μεγάλου μήκους καλωδιώσεων σύστημα, επιτρέπει την εύκολη εγκατάσταση και προσαρμογή του μέσα στο σπίτι ή ακόμα και την άμεση μεταφορά του σε κάποιο άλλο.
- Προσφέρει επεκτασιμότητα, εφόσον οι ανάγκες επιτάσσουν περισσότερους αισθητήρες. Η ευρεία γκάμα από χαμηλού κόστους αισθητήρες, συμβατούς με μικρό-ελεγκτές που υπάρχει στην αγορά, καθιστά εφικτό να προστεθούν στο σύστημα και άλλες λύσεις αυτοματισμού, με μικρή παρέμβαση κώδικα που θα είναι ωστόσο αναγκαία τόσο για να αναγνωρίζονται από αυτό όσο και για να υπάρχει προβολή των δεδομένων τους στις client εφαρμογές (web client, android app).

2.3 Σύστημα οικιακών αυτοματισμών ως ασύρματο δίκτυο αισθητήρων

Για την κάλυψη των απαιτήσεων που αναφέρθηκαν, το home automation σύστημα θα βασίζεται σε ένα ασύρματο δίκτυο αισθητήρων εσωτερικού χώρου.

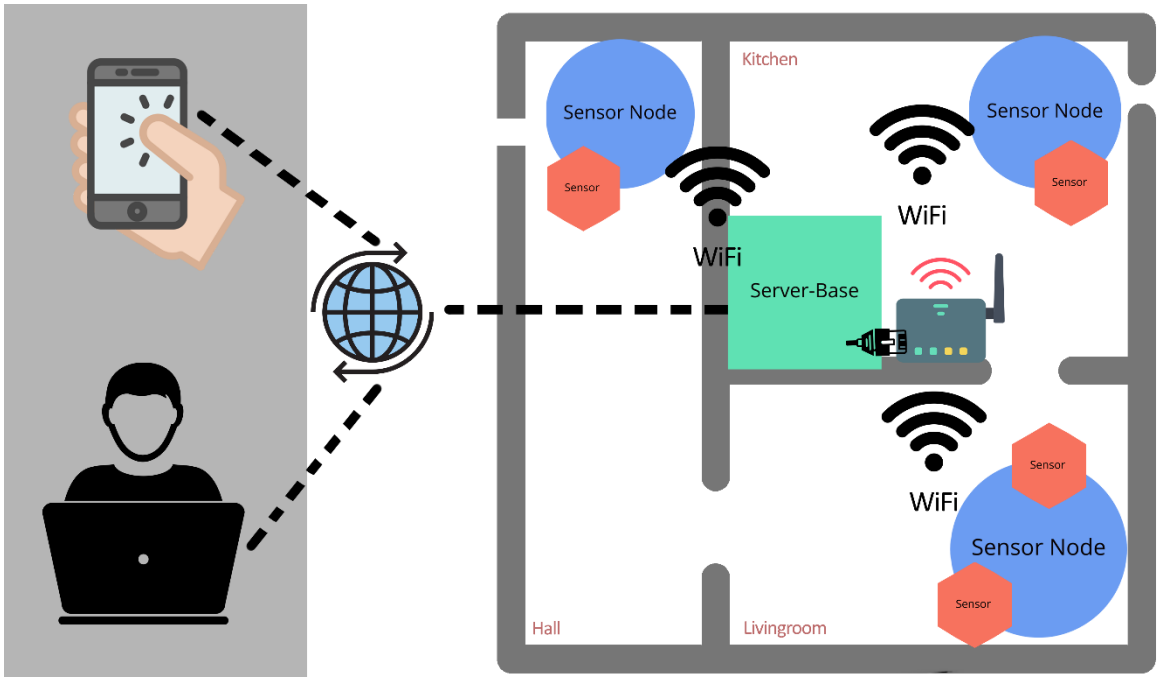
Ασύρματα δίκτυα αισθητήρων ή στα αγγλικά Wireless Sensor Networks (WSNs) μπορούν να οριστούν τα δίκτυα αυτόνομων, διασκορπισμένων κόμβων ή sensor Nodes, στα οποία είναι συνδεμένοι ένας η περισσότεροι αισθητήρες διαφόρων ειδών (θερμοκρασίας, ήχου, κίνησης κ.α.), και είναι υπεύθυνοι για την αλληλεπίδραση με το φυσικό περιβάλλον και τη συλλογή ή επεξεργασία από αυτό δεδομένων. Τα δεδομένα αυτά στη συνέχεια μεταφέρονται πάνω στο δίκτυο, συνήθως σε μια κεντρική τοποθεσία, με σκοπό την παρακολούθηση και ανάλυση τους [3].

Σύμφωνα με το [3], στα WSNs υπάρχουν περισσότερες του ενός τοπολογίες που αφορούν τη μορφολογία του δικτύου και αυτές ονομαστικά απαρτίζονται από τις δομές Star Network, Mesh Network και Hybrid Star-Mesh Network (βλέπε Σχήμα 2.1). Η τοπολογία Star Network, η οποία και θα εφαρμοστεί στο σύστημα μας, αποτελείται από μία κεντρική βάση αποκλειστικά υπεύθυνη να στείλει ή να λάβει μηνύματα από και προς τα sensor nodes, καθώς η άμεση μεταξύ τους επικοινωνία δεν υφίσταται. Με τη δομή αυτή επιτυγχάνεται η απλότητα και η ελάχιστη κατανάλωση ενέργειας στους κόμβους, αλλά επιβάλλει στους τελευταίους να βρίσκονται υποχρεωτικά στο εύρος ανίχνευσης τους από τη βάση.



Σχήμα 2.1: (Αριστερά προς τα δεξιά) - Απεικόνιση τοπολογιών Star Network, Mesh Network, Hybrid Star-Mesh Network [3]

Έτσι λοιπόν, εφαρμόζοντας ένα ασύρματο δίκτυο αισθητήρων με τη Star Network τοπολογία για το σύστημα οικιακών αυτοματισμών που υλοποιούμε, η τελική μορφή του θα είναι όπως απεικονίζεται στο Σχήμα 2.2. Μια κεντρική βάση συνδεμένη στο οικιακό router της οικείας, επικοινωνεί ασύρματα με τα sensor nodes που είναι κατανεμημένα στα δωμάτια του σπιτιού και αυτά με τη σειρά τους έχουν συνδεμένους πάνω τους τούς διάφορους αισθητήρες του συστήματος. Ο χρήστης στης συνέχεια επικοινωνεί με τη βάση μέσω των εφαρμογών που είναι διαθέσιμες στις καθημερινές συσκευές που χρησιμοποιεί.



Σχήμα 2.2: Απεικόνιση κάτοψης σπιτιού στο οποίο είναι εγκατεστημένο το σύστημα οικιακών αυτοματισμών στη μορφή ενός WSN με Star Network τοπολογία

ΚΕΦΑΛΑΙΟ 3

ΕΠΙΛΟΓΗ ΤΟΥ HARDWARE, ΠΡΟΕΤΟΙΜΑΣΙΑ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΤΩΝ SENSOR NODES

3.1 Επιλογή της βάσης του συστήματος

Για να εφαρμοστεί η δομή Star Network στο σύστημα μας, αρχικά επιλέγεται ως βάση και κεντρικός διακοσμητής (server), ένας μικρού μεγέθους, χαμηλού κόστους και χαμηλής κατανάλωσης υπολογιστής, το Raspberry Pi (βλέπε Εικόνα 3.1). Ο υπολογιστής αυτός, ως η κεντρική βάση του συστήματος, θα συνδεθεί στο router του δικτύου της οικίας (ενσύρματα με τη χρήση ενός καλωδίου Ethernet) και θα έχει το ρόλο του κεντρικό εξυπηρετητή (server). Ο server έχει την αρμοδιότητα να επικοινωνεί με τα sensor nodes πάνω από το δίκτυο, είτε συλλέγοντας τα δεδομένα των αισθητήρων είτε δίνοντας εντολές στους κόμβους και αυτοί κατόπιν στους αισθητήρες για τη πραγματοποίηση κάποιας ενέργειας (π.χ. άνοιγμα/κλείσιμο λάμπας).

Εικόνα 3.1: Απεικόνιση του Raspberry Pi

Τα μοντέλα-εκδοχές του Raspberry Pi είναι πολλά, ενώ για το σύστημα μας θα χρησιμοποιηθεί το Raspberry Pi 4 Model B 4GB (για τα αναλυτικά τεχνικά χαρακτηριστικά βλέπε [4]), ένα από τα νεότερα μοντέλα τη στιγμή που γράφεται το κείμενο. Σημειώνεται ότι εξαιτίας των χαμηλών πόρων που θα απαιτεί η εφαρμογή του server που επρόκειτο να εγκατασταθεί, παλιότερα μοντέλα του Raspberry Pi θα μπορούσαν επίσης να χρησιμοποιηθούν, χωρίς ωστόσο αυτό να έχει τεθεί υπό δοκιμή. Όσον αφορά τη τροφοδοσία της συσκευής, μπορεί να χρησιμοποιηθεί ένας κλασσικός φορτιστής κινητών τηλεφώνων, με την απαίτηση το καλώδιο να έχει απόληξη USB Type-C για να μπορεί να

εφαρμοστεί στην αντίστοιχη θύρα του συγκεκριμένου μοντέλου. Στην Εικόνα 3.2 απεικονίζεται η τοποθέτηση του Raspberry Pi στο χώρο και συγκεκριμένα η σύνδεση του τόσο στο ρεύμα όσο και στο router του δικτύου της οικίας (ενσύρματα μέσω Ethernet).

Εικόνα 3.2: Απεικόνιση του Raspberry Pi τοποθετημένο στο χώρο του σπιτιού

3.2 Αρχική προετοιμασία του Raspberry Pi

Το Raspberry Pi για να χρησιμοποιηθεί, απαιτεί μια αρχική προετοιμασία. Η διαδικασία αυτή αφορά την εγκατάσταση λειτουργικού συστήματος στη συσκευή, και την επίτευξη απομακρυσμένης σύνδεσης προς αυτή, εφόσον δεν υπάρχει η δυνατότητα καλωδίωσης της σε κάποια οθόνη και πληκτρολόγιο. Ακολουθώντας τον οδηγό [5] της επίσημης ιστοσελίδας του Raspberry Pi για την εγκατάσταση λειτουργικού συστήματος και τον αντίστοιχο οδηγό [6] για την απομακρυσμένη σύνδεση στη συσκευή, πραγματοποιούμε τις παρακάτω ενέργειες:

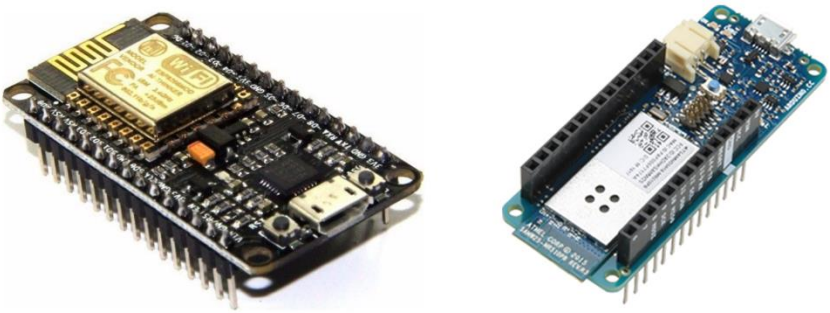
- Με τη βοήθεια ενός δεύτερου υπολογιστή, τοποθετούμε τα κατάλληλα αρχεία που απαρτίζουν το λειτουργικό Raspbian OS συν ένα άδειο αρχείο με όνομα *ssh* σε μια κάρτα SD 32GB που προορίζεται για το Raspberry Pi.
- Τοποθετούμε τη κάρτα SD στο Raspberry Pi και έπειτα συνδέουμε την συσκευή στο ρεύμα και στο router της οικίας, όπως φαίνεται στην Εικόνα 3.2.
- Η συσκευή είναι τώρα προσπελάσιμη μέσω του πρωτοκόλλου SSH. Συνδεόμαστε απομακρυσμένα σε αυτή, μέσω ενός δεύτερου υπολογιστή.

3.3 Επιλογή hardware για τα sensor nodes του συστήματος

Αφού ορίστηκε η βάση του της Star τοπολογίας του συστήματος, σειρά έχουν να επιλεχθούν τα διάφορα sensor nodes. Τα sensor nodes, σε ένα home automation σύστημα, καταμερίζονται στο σπίτι και στα διάφορα δωμάτια με βάση τις ανάγκες, τους αυτοματισμούς που χρειάζεται να επιτευχθούν και τους αισθητήρες που θα συνδεθούν σε αυτά. Απαραίτητος παράγοντας είναι να βρίσκονται μέσα στη περιοχή του οικιακού δικτύου, ώστε να έχουν πρόσβαση σε αυτό και άρα να είναι εφικτή η ασύρματη επικοινωνία τους με το Raspberry Pi. Συνεπώς, για την υλοποίηση τους απαιτούνται συσκευές με ικανότητα σύνδεσης Wi-Fi και ταυτόχρονα δυνατότητα να τρέξουν κώδικα, ώστε να παρέχεται ευελιξία στην επικοινωνία τους με διαφόρων ειδών αισθητήρες, μέσα από τον προγραμματισμό τους.

Για τον σκοπό αυτό, τα sensor nodes του συστήματος απαρτίζονται από boards μικρό-ελεγκτών. Οι μικρό-ελεγκτές (αγγλικά: microcontrollers) είναι ολοκληρωμένα κυκλώματα που τυπικά περιέχουν επεξεργαστή, μνήμη και περιφερειακά εισόδου/εξόδου και αποτελούν μέρη ενός ενσωματωμένου συστήματος που προγραμματίζεται να εκτελεί μια συγκεκριμένη λειτουργία σε επανάληψη [7].

Για τις ανάγκες του παρόντος συστήματος, οι single board microcontrollers που επιλέχθηκαν είναι δύο NodeMCU ESP8266 Wi-Fi boards και ένα Arduino MKR1000 (βλέπε Εικόνα 3.3, για τα τεχνικά χαρακτηριστικά βλέπε [8, 9]). Το NodeMCU έχει αρκετά χαμηλό κόστος, προσφέρει Wi-Fi σύνδεση (εξαιτίας του ESP8266 Wi-Fi SoC που περιέχει) και διαθέτει πολλαπλά Pins εισόδου/εξόδου και γενικού σκοπού (GPIOS). Τα pins αυτά συνήθως αξιοποιούνται μέσα από τη σύνδεση τους με αισθητήρες. Ανάλογων προδιαγραφών (πάντα με βάση τις ανάγκες του παρόντος συστήματος) microcontroller, είναι και το Arduino MKR1000 Wi-Fi. Το συγκεκριμένο board, στην παρούσα εφαρμογή χρησιμοποιείται ακριβώς όπως τα NodeMCU και επιλέχθηκε διότι υπήρχε ήδη στην κατοχή μας και δε χρειάστηκε να αγοραστεί για τις ανάγκες της διπλωματικής εργασίας.



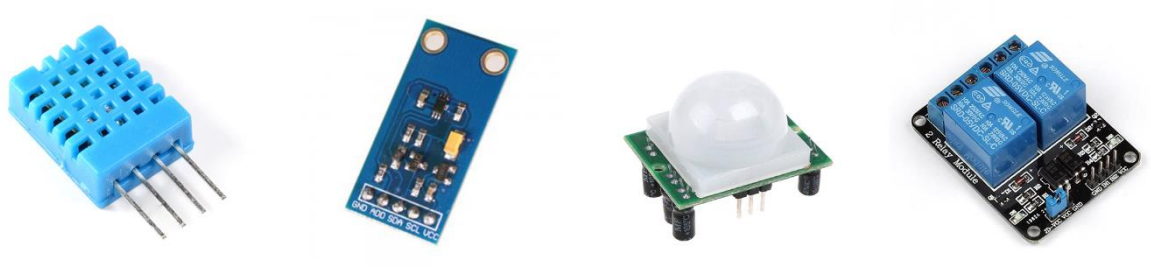
Εικόνα 3.3: Απεικόνιση NodeMCU και Arduino MKR1000 (Αριστερά προς τα δεξιά)

Αμφότερα τα είδη των microcontrollers που αναφέρθηκαν, μπορούν να λάβουν την τροφοδοσία τους από τη Micro-USB θύρα τους που είναι κολλημένη πάνω στη πλακέτα. Τονίζεται πως η ίδια θύρα χρησιμοποιείται και για το πέρασμα κώδικα στις δύο συσκευές, εφόσον αυτά έχουν συνδεθεί κατάλληλα σε έναν υπολογιστή και η διαδικασία γίνεται μέσα από τη χρήση κατάλληλου IDE (Integrated Development Environment) Software (π.χ. Arduino IDE, VS Code, κ.ά.)

3.4 Επιλογή των αισθητήρων

Οι αισθητήρες του συστήματος είναι ο βασικότερος παράγοντας που χαρακτηρίζει το είδος των λειτουργιών του. Η ποικιλία τους επιτρέπει, αφού αρχικά προσδιορίσουμε τις ανάγκες και τους οικιακούς αυτοματισμούς που θέλουμε να επιτύχουμε στο σύστημα, να επιλέξουμε τους κατάλληλους και να τους εφαρμόσουμε πάνω στα microcontroller boards που αναφέρθηκαν παραπάνω.

Για τις ανάγκες του home automation που παρουσιάζεται, επιλέχθηκαν χαμηλού κόστους αισθητήρες που είναι όμως αρκετά αξιόπιστοι για την σωστή λειτουργία των αυτοματισμών που θέλουμε να επιτύχουμε. Αυτοί είναι οι εξής (βλέπε Εικόνα 3.4): ένας αισθητήρας μέτρησης θερμοκρασίας-υγρασίας (DHT11 Temperature and Humidity Sensor Module for Arduino, για τεχνικά χαρακτηριστικά βλέπε [10]), ένας αισθητήρας ανίχνευσης κίνησης(HC-SR501 PIR Motion Sensor Module for Arduino, για τεχνικά χαρακτηριστικά βλέπε [11]), ένας αισθητήρας ανίχνευσης φωτεινότητας(BH1750FVI Digital Light Sensor Module For Arduino, για τεχνικά χαρακτηριστικά βλέπε [12]), και ένα 2-καναλιών ρελέ (3.3V 2-Channel Relay Module Board for Arduino / ESP8266, για τεχνικά χαρακτηριστικά βλέπε [13]).



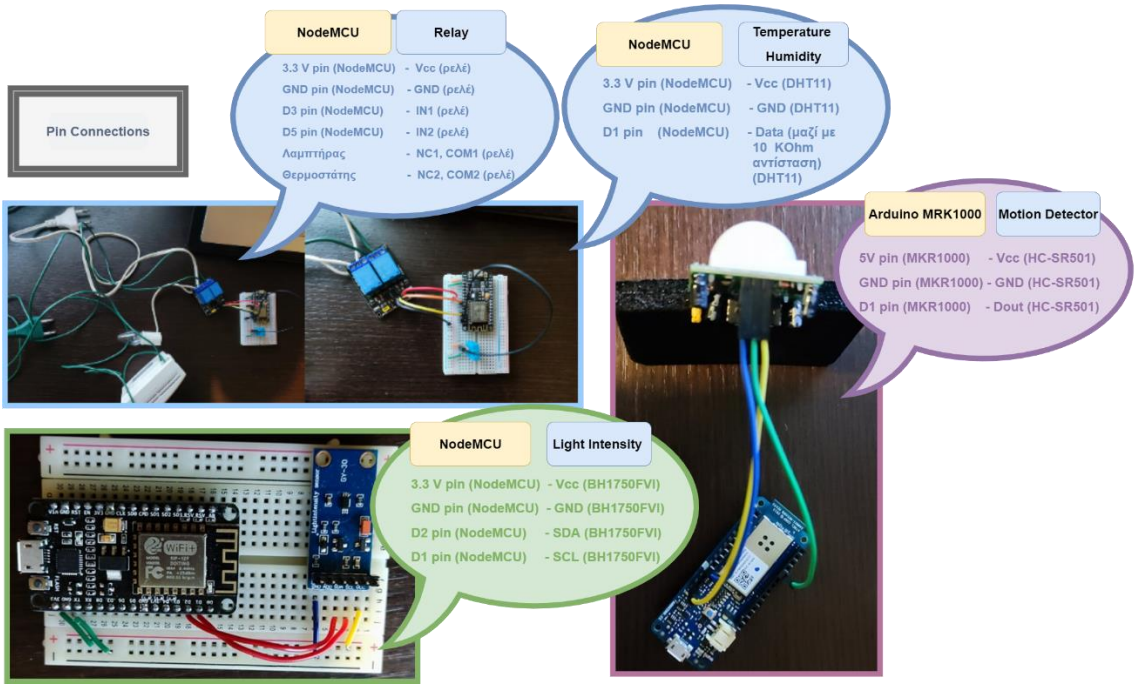
Εικόνα 3.4: (Αριστερά προς τα δεξιά) - Απεικόνιση αισθητήρα μέτρησης θερμοκρασίας-υγρασίας DHT11, αισθητήρα ανίχνευσης φωτεινότητας BH1750FVI, αισθητήρα ανίχνευσης κίνησης HC-SR501 και ρελέ 2-καναλιών 3.3V.

Με τη συγκεκριμένη επιλογή αισθητήρων δημιουργείται μια πληθώρα επιλογών σε αυτοματισμούς και εύχρηστες λειτουργίες. Όπως και θα δειχτεί παρακάτω, με την εγκατάσταση του συστήματος, ο χρήστης έχει τη δυνατότητα, χάρις τους διαθέσιμους αισθητήρες, να θέσει αυτοματισμούς όπως το άνοιγμα της θέρμανσης (συνδεμένη στο ρελέ) όταν πέσει η θερμοκρασία, το άνοιγμα της λάμπας ενός δωματίου (ομοίως συνδεμένη στο ρελέ) όταν σκοτεινιάσει η μέρα ή όταν ανιχνευθεί κίνηση κ.ά. Σε περίπτωση που οι ανάγκες αυξηθούν, μπορούν πάντα να αγοραστούν περισσότεροι αισθητήρες και να προστεθούν στο σύστημα, και αυτό διότι το κόστος αγοράς τους βρίσκεται σε χαμηλά επίπεδα.

3.5 Κατασκευή των sensor nodes

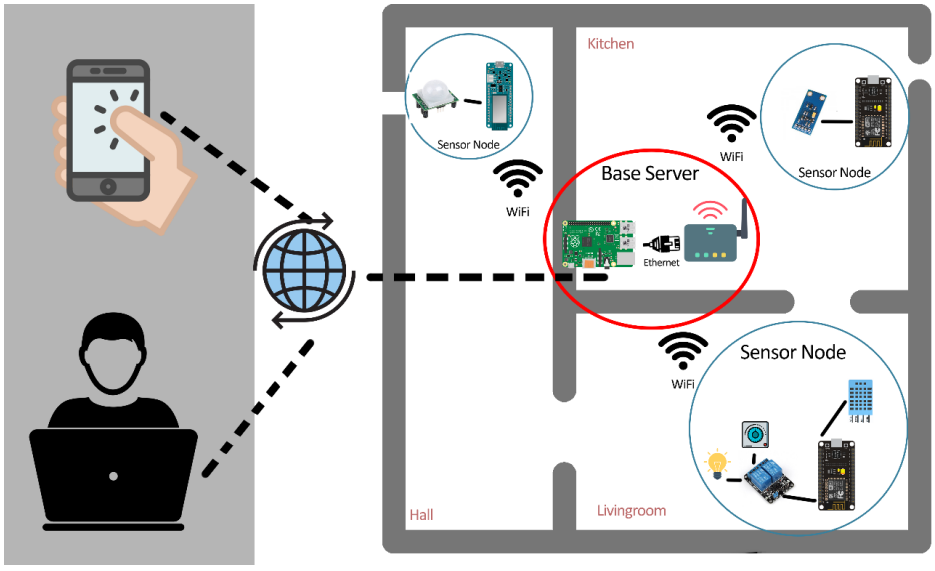
Η κατανομή των επιλεγμένων αισθητήρων στους διαθέσιμους microcontrollers (τα δύο NodeMCU και το ένα Arduino MKR1000) είναι ελεύθερη επιλογή του χρήστη. Συνεπώς, στο σύστημα μας επιλέγουμε να συνδέσουμε τον αισθητήρα θερμοκρασίας-υγρασίας και το ρελέ 2-καναλιών στο ένα NodeMCU και παράλληλα στο άλλο τοποθετούμε τον αισθητήρα ανίχνευσης έντασης φωτεινότητας. Επιπλέον, στο Arduino MKR1000 καλωδιώνεται επάνω του ο αισθητήρας ανίχνευσης κίνησης, ενώ στα δυο κανάλια του ρελέ συνδέουμε από μια λάμπα και έναν θερμοστάτη. Συμβουλεύοντας τα τεχνικά χαρακτηριστικά των microcontroller boards [8,9] και των αισθητήρων [10,11,12,13], τα διασυνδέουμε με τη βοήθεια καλωδίων. Στο Σχήμα 3.1 φαίνεται αναλυτικά η αντιστοίχιση των pins τους.

Σημείωση: Για την αποφυγή μεγάλων σχετικά με τα εξαρτήματα καλωδίων και για καλύτερη οργάνωση, χρησιμοποιούνται breadboards.



Σχήμα 3.1: Απεικόνιση της συνδεσμολογίας των pins μεταξύ αισθητήρων και microcontrollers

Τοποθετώντας τα υλοποιημένα sensor nodes στους διάφορους χώρους του σπιτιού, το σχεδιάγραμμα του συστήματος αυτοματισμών μπορεί να γίνει πιο συγκεκριμένο και φαίνεται στο Σχήμα 3.2.



Σχήμα 3.2: Απεικόνιση κάτοψης σπιτιού στο οποίο είναι εγκατεστημένο το σύστημα αυτοματισμών χρησιμοποιώντας το Raspberry Pi ως βάση και τα sensor nodes που υλοποιήθηκαν στην Ενότητα 3.5.

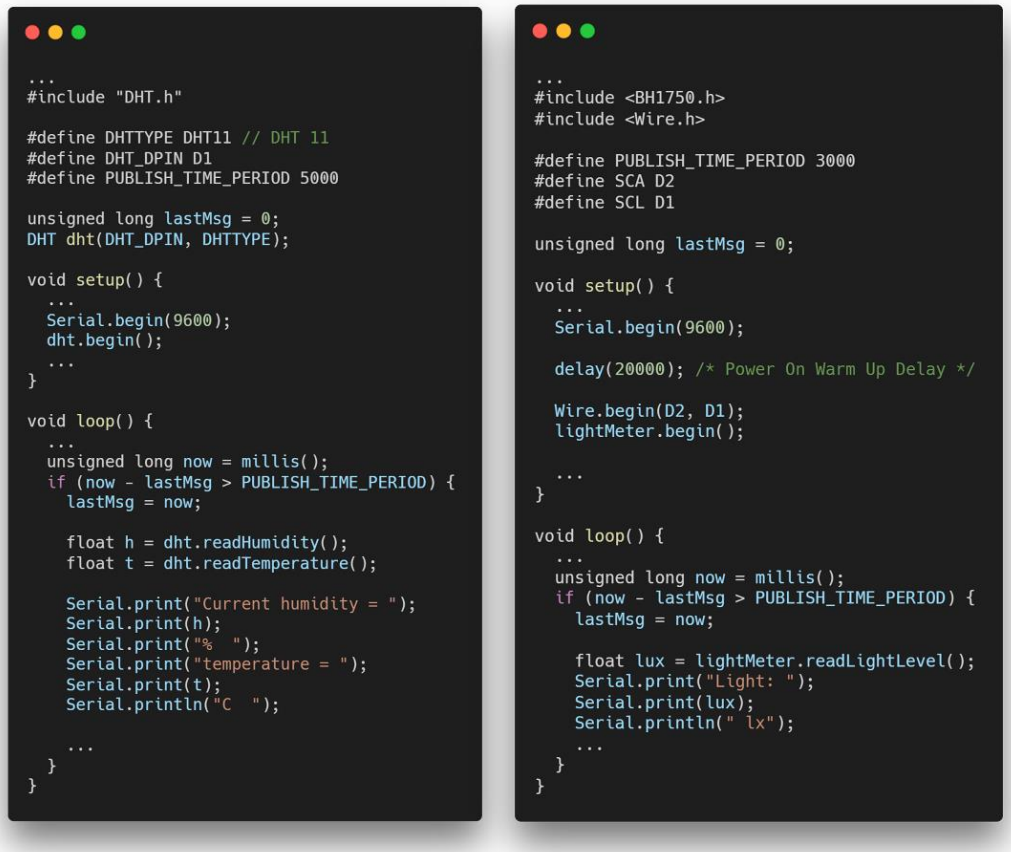
3.6 Προγραμματισμός των microcontroller boards

Αφού κατασκευάστηκαν, για να γίνουν στη συνέχεια λειτουργικά τα sensor nodes, πρέπει να στείλουμε κατάλληλο κώδικα μνήμης flash στους μικρό-ελεγκτές (με τη χρήση του κατάλληλου IDE και μέσω καλωδίου USB). Επιπλέον, τα pins που έχουν πάνω τους καλωδιωμένους τούς διάφορους αισθητήρες που αναφέρθηκαν, μπορούν να γίνουν προσπελάσιμα στον κώδικα χρησιμοποιώντας τα κατάλληλα σύμβολα (π.χ. D1, D2).

Η γλώσσα προγραμματισμού που χρησιμοποιείται για να συντελέσει τον κώδικα που στέλνεται στα boards ονομάζεται Arduino Programming language και έχει πολλές ομοιότητες με τη δομή της γλώσσας C++. Επιπλέον, ο κώδικας που εκτελείται σε κάθε board διαθέτει πάντα τις δυο βασικές συναρτήσεις setup() και loop(). Η συνάρτηση setup() εκτελείται μια φορά με κάθε εκκίνηση της συσκευής, ενώ η συνάρτηση loop() έπεται της setup() και η εκτέλεση της επαναλαμβάνεται επ' άπειρον, εφόσον υφίσταται τροφοδοσία ρεύματος στο board.

Όσον αφορά την αξιοποίηση των αισθητήρων που επιλέχθηκαν, υπάρχει πληθώρα open-source Arduino βιβλιοθηκών που διευκολύνουν την προσπέλαση τους μέσα από τον κώδικα της εφαρμογής και κατόπιν τη συλλογή δεδομένων από αυτούς.

Έτσι, μέσω των βιβλιοθηκών adafruit/DHT-sensor-library [14] και claws/BH1750 [15], γίνεται εύκολη η συγκέντρωση πληροφορίας για τους αισθητήρες θερμοκρασίας-υγρασίας(DHT11) και φωτεινότητας(BH1750FVI) αντίστοιχα. Στο Σχήμα 3.3 μπορεί να φανεί πως με τη χρήση της γλώσσας Arduino Programming Language και των δυο αυτών βιβλιοθηκών, είναι εφικτή η ανάγνωση της πληροφορίας από τα boards. Συγκεκριμένα, για να διαβαστεί η θερμοκρασία και η υγρασία χρησιμοποιούνται αντίστοιχα οι συναρτήσεις dht.readTemperature() και dht.readHumidity() (αριστερό πλαίσιο κώδικα), ενώ για το διάβασμα της έντασης φωτεινότητας γίνεται χρήση της lightMeter.readLightLevel() (δεξί πλαίσιο κώδικα). Σημειώνεται ότι για τους υπόλοιπους αισθητήρες δε χρειάστηκε κάποια πρόσθετη βιβλιοθήκη για να διαχειριστούμε τα δεδομένα τους, αφού οι built-in συναρτήσεις που προσφέρει η Arduino Programming Language ήταν αρκετές.



```
...
#include "DHT.h"

#define DHTTYPE DHT11 // DHT 11
#define DHT_DPIN D1
#define PUBLISH_TIME_PERIOD 5000

unsigned long lastMsg = 0;
DHT dht(DHT_DPIN, DHTTYPE);

void setup() {
  ...
  Serial.begin(9600);
  dht.begin();
  ...
}

void loop() {
  ...
  unsigned long now = millis();
  if (now - lastMsg > PUBLISH_TIME_PERIOD) {
    lastMsg = now;

    float h = dht.readHumidity();
    float t = dht.readTemperature();

    Serial.print("Current humidity = ");
    Serial.print(h);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(t);
    Serial.println("C  ");

    ...
  }
}
```

```
...
#include <BH1750.h>
#include <Wire.h>

#define PUBLISH_TIME_PERIOD 3000
#define SCA D2
#define SCL D1

unsigned long lastMsg = 0;

void setup() {
  ...
  Serial.begin(9600);

  delay(20000); /* Power On Warm Up Delay */

  Wire.begin(D2, D1);
  lightMeter.begin();

  ...
}

void loop() {
  ...
  unsigned long now = millis();
  if (now - lastMsg > PUBLISH_TIME_PERIOD) {
    lastMsg = now;

    float lux = lightMeter.readLightLevel();
    Serial.print("Light: ");
    Serial.print(lux);
    Serial.println(" lx");
    ...
  }
}
```

Σχήμα 3.3: Απεικόνιση μέρους του κώδικα που στέλνεται στα boards και αφορά τη συλλογή δεδομένων από τα pins που είναι συνδεμένοι οι αισθητήρες. Το πλαίσιο αριστερά αφορά τον αισθητήρα DHT11, ενώ το πλαίσιο δεξιά, τον αισθητήρα BH1750FVI.

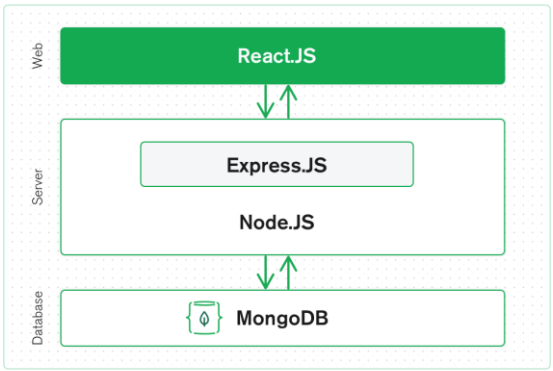
ΚΕΦΑΛΑΙΟ 4

ΕΠΙΛΟΓΗ ΤΩΝ SOFTWARE ΕΡΓΑΛΕΙΩΝ, ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ SERVER ΚΑΙ ΣΥΝΔΕΣΗ ΜΕ ΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

4.1 Προσδιορισμός του software για τον server της εφαρμογής

Μετά την επιλογή του hardware, υπολείπεται να προσδιοριστούν τα software εργαλεία που θα χρησιμοποιηθούν. Η φύση του συστήματος οικιακών αυτοματισμών επιβάλλει από το software την ύπαρξη ενός κεντρικού server για τη διαρκή επικοινωνία του με τα sensor nodes, μιας βάσης δεδομένων για την αναγκαία αποθήκευση δεδομένων και μια client εφαρμογή για την επικοινωνία του χρήστη με τον server και κατά επέκταση την αλληλεπίδραση του με το σύστημα.

Με κύριο εργαλείο τη γλώσσα προγραμματισμού JavaScript, ο backend τομέας του software θα απαρτίζεται από το Node(.js) και μαζί με το Express(.js) web framework θα αποτελέσουν τη πλατφόρμα του server. Η MongoDB θα χρησιμοποιηθεί ως η NoSQL βάση δεδομένων της εφαρμογής, ενώ μέσα από το React(.js) θα υλοποιηθεί το frontend, δηλαδή ένας δυναμικός Web Client συμβατός τόσο με desktop όσο και με mobile browsers. Το σύνολο των τεχνολογικών εργαλείων αυτών αποτελεί τη δημοφιλή MERN(Mongo-Express-React-Node) Stack (βλέπε ΣΑΧήμα 4.1) και αποτελεί ισχυρό εργαλείο για full stack εφαρμογές, βασισμένες στη γλώσσα JavaScript [16].



Σχήμα 4.1: Απεικόνιση των επιπέδων της στοίβας MERN [16].

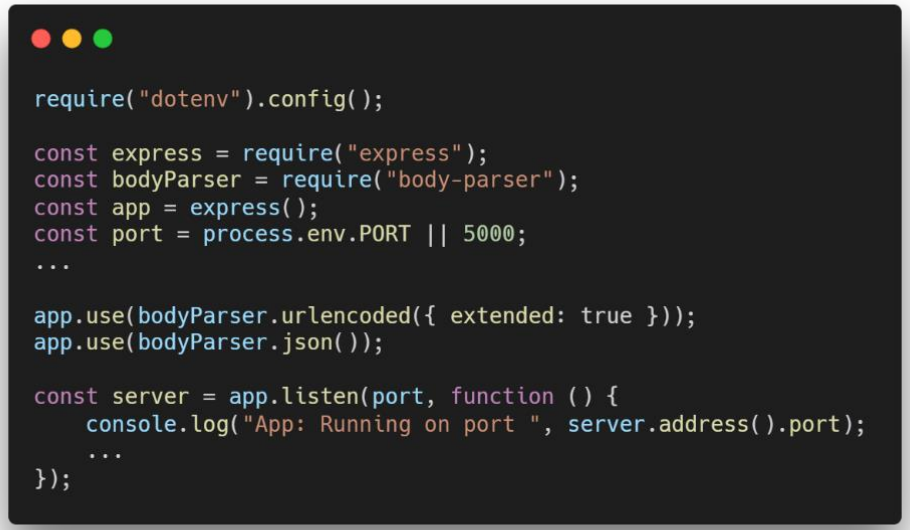
Το σύστημα θα έχει ακόμη και έναν Android client, πέραν του αντίστοιχου web. Όπως θα αναλυθεί και στη συνέχεια, ο React web client θα μετατραπεί σε εφαρμογή Android, μέσα

από το εργαλείο Apache Cordova, το οποίο είναι μια τεχνολογία που μπορεί να μετατρέψει μια web εφαρμογή HTML/CSS/JavaScript σε native εφαρμογή για κινητά τηλέφωνα.

4.2 Δημιουργία του server μέσω Node

Είναι πολύ απλό να κατασκευάσουμε έναν server μέσω Node. Εγκαθιστώντας δωρεάν το Node(.js) και το npm package manager, το οποίο διαθέτει ένα τεράστιο πλήθος πακέτων με JavaScript βιβλιοθήκες, κάνουμε στη συνέχεια τα εξής:

- Προσθέτουμε και εγκαθιστούμε στην εφαρμογή, μέσω του npm, τα modules express (το web framework που αναφέρθηκε προηγουμένως), body-parser (parser που μετατρέπει τα δεδομένα των HTTP requests σε JavaScript objects) και το dotenv (parser αρχείων .env για την επεξεργασία τους στο κώδικα).
- Με τη βοήθεια των παραπάνω βιβλιοθηκών, όπως φαίνεται στο Σχήμα 4.2, ορίζουμε τον server να «τρέχει» και να «ακούει» στη πόρτα 5000. Για την έναρξη της εφαρμογής τρέχουμε σε έναν τερματικό την εντολή node server.js, όπου server.js, το κύριο αρχείο της εφαρμογής, το οποίο περιέχει και τον σχετικό κώδικα του Σχήματος 4.2.



Σχήμα 4.2: Απεικόνιση μέρους του κώδικα για την κατασκευή και εκκίνηση του server σε Node

4.3 Εγκατάσταση και σύνδεση της MongoDB

Μέσω της MongoDB, υπάρχει η επιλογή της κατασκευής μιας βάσης δεδομένων είτε στο cloud, είτε τοπικά σε κάποιον υπολογιστή. Όσον αφορά την πρώτη επιλογή, η υπηρεσία MongoDB Atlas προσφέρει μια δωρεάν επιλογή δημιουργίας της βάσης δεδομένων στο cloud, με χώρο ελεύθερο προς δέσμευση, από 500 MB ως 5 GB. Από την άλλη πλευρά, για την εγκατάσταση της βάσης τοπικά, υπάρχει διαθέσιμη η υπηρεσία του MongoDB Community Server[17].

Στο σύστημα οικιακών αυτοματισμών που κατασκευάζεται και δεδομένων των ελαχίστων πόρων που καταναλώνει ως μια ατομική εφαρμογή, αμφότερες οι δυο επιλογές είναι κατάλληλες. Μολονότι, η λογική της εγκατάστασης της βάσης τοπικά, εναρμονίζεται με την γενικότερη φύση του συστήματος που δίνει στον χρήστη τον απόλυτο έλεγχο, τόσο στο hardware που διαθέτει όσο και στον server που τρέχει πάνω σε αυτό (ο server τρέχει τοπικά στο Raspberry Pi, χωρίς να γίνεται deploy σε κάποια cloud υπηρεσία), η cloud πλατφόρμα της MongoDB προσφέρει από την άλλη τα εξής προτερήματα στο σύστημα:

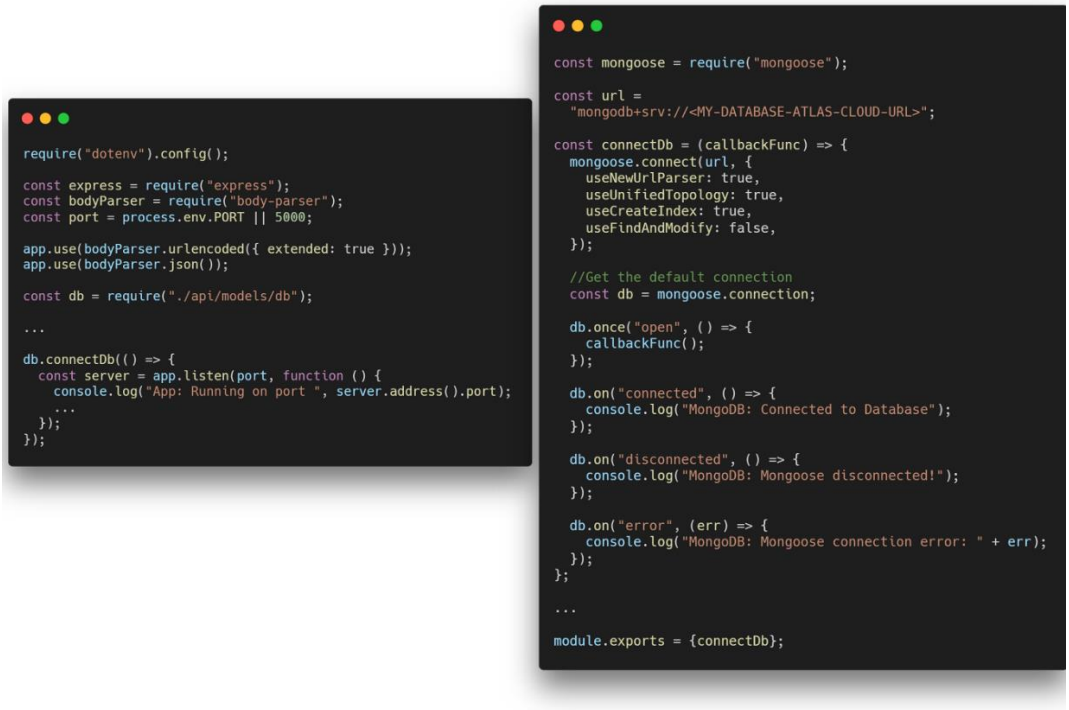
- Αποφεύγεται ο κίνδυνος καταστροφής των δεδομένων (περίπτωση καταστροφής της SD Card του Raspberry Pi).
- Προσφέρει στον χρήστη απομακρυσμένη πρόσβαση στα δεδομένα του συστήματος από οποιαδήποτε συσκευή και σε οποιοδήποτε δίκτυο, και ταυτόχρονα κάνει εύκολη τη παρακολούθηση τους, μέσω του γραφικού περιβάλλοντος της ιστοσελίδας του MongoDB Atlas.

Συνεπώς, για το σύστημα μας προτιμάται η cloud υπηρεσία MongoDB Atlas έναντι της τοπικής λύσης. Όπως επισημάνθηκε και πριν, ο χώρος που προσφέρει η υπηρεσία είναι άφθονος για τις ανάγκες του συστήματος, αλλά εφόσον χρειαστεί, παραμένει απλή και εφικτή η μεταφορά της βάσης από το cloud στο διαθέσιμο τοπικό χώρο του Raspberry Pi.

Όσον αφορά τον τρόπο χρήσης της βάσης στο σύστημα, θα αναλυθεί στη συνέχεια πως αυτός σχετίζεται με την αποθήκευση της πληροφορίας εκείνης που προέρχεται από αισθητήρες για τους οποίους είναι απαραίτητη η καταγραφή μετρήσεων σε συγκεκριμένα διαστήματα του χρόνου (π.χ. μετρήσεις θερμοκρασίας που έγιναν χθες ανάμεσα στις 8 μ.μ. και 9 μ.μ.). Επιπρόσθετα, η βάση αξιοποιείται και για την αποθήκευση των αυτοματισμών που ορίζει ο χρήστης (π.χ. άνοιγμα του λαμπτήρα όταν σκοτεινιάζει) μέσα

από τις client εφαρμογές. Με αυτόν τον τρόπο, διασφαλίζεται ότι αμφότερα τα δεδομένα και οι ρυθμίσεις του χρήστη δεν θα χαθούν με επανεκκίνηση του Raspberry Pi άρα και ολόκληρου του συστήματος.

Για την εγκατάσταση της βάσης και τη χρήση της στη Node εφαρμογή, εγκαθιστούμε μέσω του npm package manager, το Mongoose module. Η Mongoose είναι μια ODM (Object Data Modeling) βιβλιοθήκη και βοηθάει στη κατασκευή μοντέλων για τα δεδομένα της βάσης, άλλα και για την σωστή επαλήθευση τους κατά τη χρήση της στον κώδικα.
Στο Σχήμα 4.3 παρουσιάζεται ο σχετικός κώδικας που συνδέει τη βάση με την εφαρμογή. Σε ένα αρχείο db.js υλοποιούμε μια συνάρτηση connectDb (δεξί πλαίσιο), στην οποία ορίζουμε την εκκίνηση της σύνδεσης με τη βάση (αφού πρώτα έχουμε ορίσει τη βάση στην ιστοσελίδα του MongoDB Atlas) και τα αντίστοιχα events σε περίπτωση σφάλματος ή ομαλής αποσύνδεσης της, κατά τον τερματισμό της εφαρμογής. Επιπλέον, τροποποιούμε τον κώδικα του server ο οποίος εκκινεί πλέον ύστερα της εκτέλεσης της connectDb και άρα την επίτευξη σύνδεσης με τη βάση (αριστερό πλαίσιο στο Σχήμα 4.3).



```
require("dotenv").config();

const express = require("express");
const bodyParser = require("body-parser");
const port = process.env.PORT || 5000;

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const db = require("./api/models/db");

...

db.connectDb(() => {
  const server = app.listen(port, function () {
    console.log("App: Running on port ", server.address().port);
    ...
  });
});
```

```
const mongoose = require("mongoose");

const url =
  "mongodb+srv://<MY-DATABASE-ATLAS-CLOUD-URL>";

const connectDb = (callbackFunc) => {
  mongoose.connect(url, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
    useFindAndModify: false,
  });

  //Get the default connection
  const db = mongoose.connection;

  db.once("open", () => {
    callbackFunc();
  });

  db.on("connected", () => {
    console.log("MongoDB: Connected to Database");
  });

  db.on("disconnected", () => {
    console.log("MongoDB: Mongoose disconnected!");
  });

  db.on("error", (err) => {
    console.log("MongoDB: Mongoose connection error: " + err);
  });
};

...

module.exports = {connectDb};
```

Σχήμα 4.3: (Αριστερά) Τροποποίηση του κώδικα του server για να εκκινεί μετά την επίτευξη σύνδεσης με τη βάση. (Δεξιά) Υλοποίηση συνάρτησης για τη σύνδεση της εφαρμογής με τη βάση δεδομένων.

ΚΕΦΑΛΑΙΟ 5

MQTT ΠΡΩΤΟΚΟΛΛΟ

5.1 Ορισμός του MQTT πρωτοκόλλου

Στο σύστημα μας, ο κορμός της αμφίδρομης και πραγματικού χρόνου επικοινωνίας για τη συλλογή δεδομένων και των ορισμό εντολών μεταξύ server και sensor nodes βασίζεται στο MQTT πρωτόκολλο.

Το MQTT αποτελεί OASIS Standard στην επικοινωνία IoT συσκευών. Είναι ένα ελαφρύ από πόρους και απλό στη δομή του πρωτόκολλο ανταλλαγής μηνυμάτων που βασίζεται στη λογική της δημοσίευσης/εγγραφής (στα αγγλικά: publish/subscribe) και είναι κατάλληλο για δίκτυα μεταξύ συσκευών που μπορεί να είναι αναξιόπιστα, να έχουν υψηλή καθυστέρηση ή μικρό bandwidth. Στόχος του πρωτοκόλλου είναι να διασφαλίζει τη δέσμευση ελάχιστου bandwidth από πλευράς δικτύου και ελάχιστων πόρων από πλευράς συσκευών, ενώ παράλληλα στοχεύει στην αξιόπιστη μετάδοση μηνυμάτων [18].

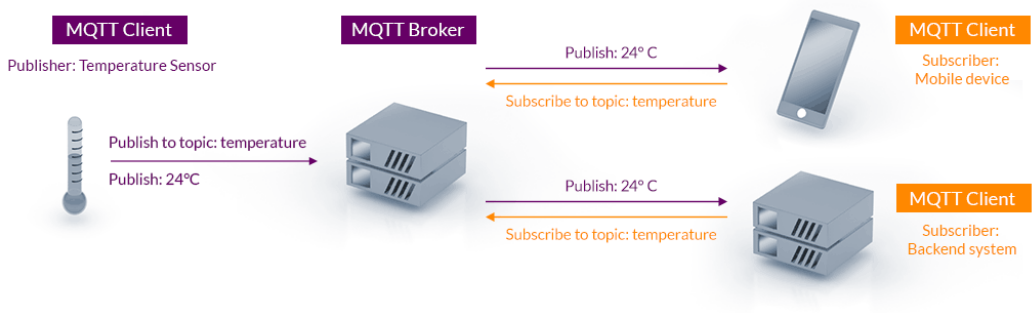
5.2 Ο τρόπος λειτουργίας του MQTT

Όπως αναφέρθηκε στην ενότητα 5.1, το MQTT στηρίζεται στο μοντέλο publish/subscribe. Στη βάση αυτού του μοντέλου, υπάρχει ένας κεντρικός broker και μαζί με αυτόν οι διάφοροι clients, οι οποίοι μπορεί να είναι είτε publishers είτε subscribers είτε και τα δυο μαζί, πάνω σε ένα συγκεκριμένο topic. Αναλυτικά, όταν ένας publisher αναλαμβάνει να στείλει μήνυμα σε ένα topic, μόνο όσοι είναι subscribers σε αυτό λαμβάνουν το μήνυμα, ενώ ο broker ως ο κεντρικός «server» είναι υπεύθυνος για να κατανέμει τα μηνύματα σε αυτούς. Δεν υπάρχει άμεση επικοινωνία μεταξύ ενός client – publisher και ενός client – subscriber, καθώς μεσολαβεί ο broker στη μετάδοση των μηνυμάτων. Στο Σχήμα 5.1 φαίνεται το σχετικό διάγραμμα της λογικής πίσω από την λειτουργία του πρωτοκόλλου, ενώ βασικά στοιχεία που συμβάλλουν στην περαιτέρω κατανόηση του είναι τα εξής [19]:

- Το πρωτόκολλο λειτουργεί πάνω από το TCP/IP για τη σύνδεση των clients με τον broker, συνεπώς χαρακτηρίζεται ως connection oriented με error correction και δέσμευση της σωστής σειράς των πακέτων κατά τη παραλαβή.
- Τα μηνύματα δε στέλνονται απευθείας από έναν client σε κάποιον άλλον με τη λογική αντιστοίχισης τους σε κάποια διεύθυνση του δεύτερου, άλλα γίνονται

publish στον broker σε ένα topic. Στη συνέχεια, ο broker είναι αυτός που αναλαμβάνει να στείλει το μήνυμα στους clients που είναι subscribers στο συγκεκριμένο topic.

- Όλοι οι clients μπορούν να κάνουν publish μηνύματα σε διάφορα topics ή να λάβουν μηνύματα από τα topics που έχουν κάνει subscribe.
- Όλοι οι clients έχουν ένα μοναδικό id, απαραίτητο μόνο στον broker στον οποίον συνδέονται, ώστε να μπορεί να τους καταχωρεί.



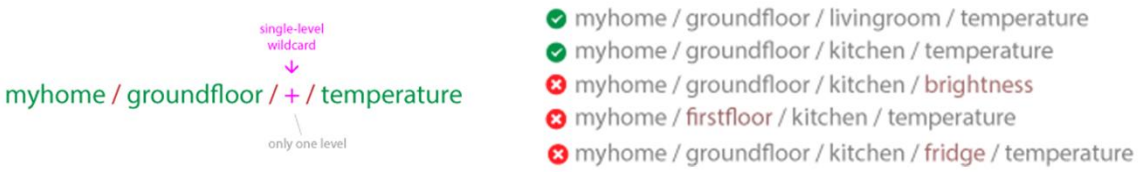
Σχήμα 5.1: Απεικόνιση της λογικής πίσω από την επικοινωνία στο MQTT [18].

Όσον αφορά την έννοια του topic που επισημαίνεται παραπάνω, ως topic στο MQTT ορίζεται ένα UTF-8 string στο οποίο δεν επιτρέπονται κενά, ενώ παράλληλα αυτό είναι case-sensitive. Όπως φαίνεται και στο Σχήμα 5.2 ένα topic μπορεί να έχει τα λεγόμενα topic levels που είναι substrings στο string του topic και χωρίζονται με τον χαρακτήρα «/» [20].

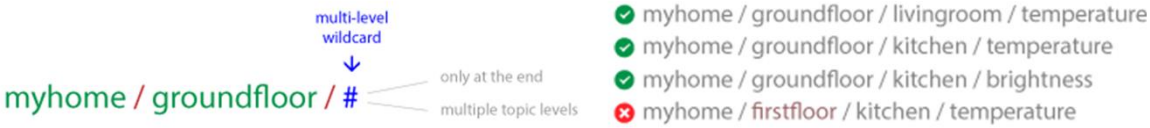
Σχήμα 5.2: Απεικόνιση ενός παραδείγματος που δείχνει τα βασικά στοιχεία ενός MQTT topic [20]

Ακόμη, στα topic levels επιτρέπονται αντί για κάποιο όνομα, τα σύμβολα «#», «+» που θεωρούνται wildcards. Το σύμβολο «+» ορίζεται ως μονού επιπέδου wildcard και μπορεί να αντικαταστήσει μόνο ένα topic level, ενώ το σύμβολο «#» θεωρείται πολύ-επίπεδο, διότι μπορεί να αντικαταστήσει πολλά επίπεδα. Με τον όρος της «αντικατάστασης» που αναφέρθηκε, εννοείται ότι όταν για παράδειγμα ένας client είναι subscriber σε ένα topic

«level1/level2/+/level3», τότε η εγγραφή του συμπεριλαμβάνει όλα τα topics που είναι πανομοιότυπα με αυτό, αλλά στη θέση του «+» έχουν κάποιο άλλο substring. Η χρήση των wildcards γίνεται καλύτερα κατανοητή και φαίνεται με λεπτομερή παραδείγματα στα Σχήματα 5.3 και 5.4.



Σχήμα 5.3: Απεικόνιση της χρήσης του single topic level wildcard «+» με παραδείγματα [20]



Σχήμα 5.4: Απεικόνιση της χρήσης του multi topic level wildcard «#» με παραδείγματα [20]

ΚΕΦΑΛΑΙΟ 6

ΜΕΤΑΔΟΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

6.1 Εισαγωγή

Έχοντας γίνει κατανοητό το MQTT Πρωτόκολλο, είναι σε αυτό το σημείο απαραίτητο να επισημανθεί ο τρόπος χρήσης του στη παρούσα εφαρμογή. Με τη χρήση κατάλληλων βιβλιοθηκών σε επίπεδο κώδικα, μπορεί να επιτευχθεί η αμφίδρομη επικοινωνία και η ανταλλαγή δεδομένων μεταξύ του Raspberry Pi και των sensor nodes σε πραγματικό χρόνο, όπου το πρώτο θα αποτελέσει τον MQTT Broker, ενώ τα sensor nodes θα λειτουργήσουν ως οι MQTT Clients.

6.2 Δημιουργία του broker στην Node εφαρμογή του server

Για να ορίσουμε το Raspberry Pi να λειτουργεί ως MQTT Broker, θα πρέπει να προσθέσουμε τη λειτουργικότητα αυτή στον κώδικα του server. Τη δυνατότητα αυτή προσφέρει η open-source JavaScript βιβλιοθήκη moscajs/aedes [21], μέσω του API (Application Programming Interface) της οποίας, μπορούμε να κατασκευάσουμε έναν broker στη Node εφαρμογή και να ορίσουμε συναρτήσεις που αφορούν την σύνδεση/αποσύνδεση των clients, το publish μηνυμάτων ή το subscribe του ίδιου του broker σε κάποιο topic για τη καταγραφή μηνυμάτων. Στο Σχήμα 6.1 παρουσιάζεται η δημιουργία του aedes broker, ο οποίος, αφού αρχικοποιηθεί, «ακούει» στη πόρτα 1883.

```
const aedes = require("aedes")();
const server = require("net").createServer(aedes.handle);
...
const port = 1883;
...

const connect = () => {

  server.listen(port, () => {
    console.log(
      "Broker: Aedes MQTT Server started and listening on port ",
      port
    );
  });

  ...

};
```

Σχήμα 6.1: Απεικόνιση του ορισμού του broker στην Node εφαρμογή του server

6.3 Ορισμός των sensor nodes ως MQTT Clients

Για να κάνουμε τους microcontrollers και κατά επέκταση τα sensor nodes να συνδεθούν στο broker της Node εφαρμογής, χρησιμοποιούμε και εδώ μια open-source βιβλιοθήκη με όνομα PubSubClient, η οποία προσφέρει τη λειτουργία του MQTT Client για Arduino εφαρμογές [22]. Μέσω αυτής, μετατρέπουμε τους διαθέσιμους microcontrollers σε clients που συνδέονται πρώτα στον broker και έπειτα, μπορούν να κάνουν publish μηνύματα ή subscribe για να λάβουν μηνύματα, σε κάποιο topic. Στο Σχήμα 6.2 φαίνεται ο σχετικός κώδικας που αφορά το NodeMCU, το οποίο έχει συνδεμένο πάνω του τον αισθητήρα μέτρησης φωτεινότητας και επιβεβαιώνει την παραπάνω λειτουργία. Στη setup συνάρτηση του board, αφού επιτευχθεί η σύνδεση στο οικιακό Wi-Fi, στη συνέχεια, το board επιχειρεί τη σύνδεση στον broker του συστήματος (συγκεκριμένα στην IP του Raspberry Pi και στη πόρτα 1883 και με ένα μοναδικό client id). Επιπλέον, στον κώδικα απεικονίζονται τα διάφορα topics που αφορούν το συγκεκριμένο sensor node, ωστόσο για αυτά θα γίνει αναφορά στην επόμενη ενότητα.

```
...
#include <PubSubClient.h> // MQTT Library
#include <ESP8266WiFi.h>  // Esp8266/NodeMCU Library

const char *ssid = "<HOME_NETWORK_SSID>";
const char *password = "<WIFI_PASSWORD>";

const char *mqtt_server = "<RASPBERRY_PI_STATIC_IP>";
const char *topicLightIntensity = "kitchen/ktn-NodeMCU/Light-Intensity/bh1750fvi";
const char *clientID = "ktn-NodeMCU"; // The client id identifies the NodeMCU device.
const char *willTopic = "kitchen/ktn-NodeMCU/device"; // Topic Status
const char *willMessage = "offline";
char buffer[256];
int willQoS = 0;
boolean willRetain = true;
WiFiClient wifiClient;
PubSubClient client(wifiClient); // 1883 is the listener port for the Broker
...

void setup() {

  Serial.begin(9600);
  setup_wifi();

  client.setServer(mqtt_server, 1883);

  StaticJsonDocument<256> doc;
  JsonArray sensors = doc.createNestedArray("sensors");
  JsonObject lightIntSensor = sensors.createNestedObject();
  lightIntSensor["type"] = "Light-Intensity";
  lightIntSensor["name"] = "bh1750fvi";

  serializeJson(doc, buffer);

  if (client.connect(clientID, "", "", willTopic, willQoS, willRetain,
                     willMessage, true)) {
    // Connecting to MQTT Broker
    client.publish(willTopic, buffer, true);

    Serial.print(clientID);
    Serial.println(" connected to MQTT Broker!");
  } else {
    Serial.print(clientID);
    Serial.println(" connection to MQTT Broker failed...");
  }

  ...
}
...
```

Σχήμα 6.2: Ο κώδικας για τον ορισμό ενός από τα sensor nodes ως MQTT Client

6.4 Τα MQTT Topics για τη σύνδεση/αποσύνδεση των sensor nodes

Τα MQTT Topics του συστήματος είναι η μοναδική δίοδος για την μετάδοση δεδομένων και παράλληλα ο κύριος παράγοντας που συμβάλει στην οργάνωση των δεδομένων που ανταλλάσσονται.

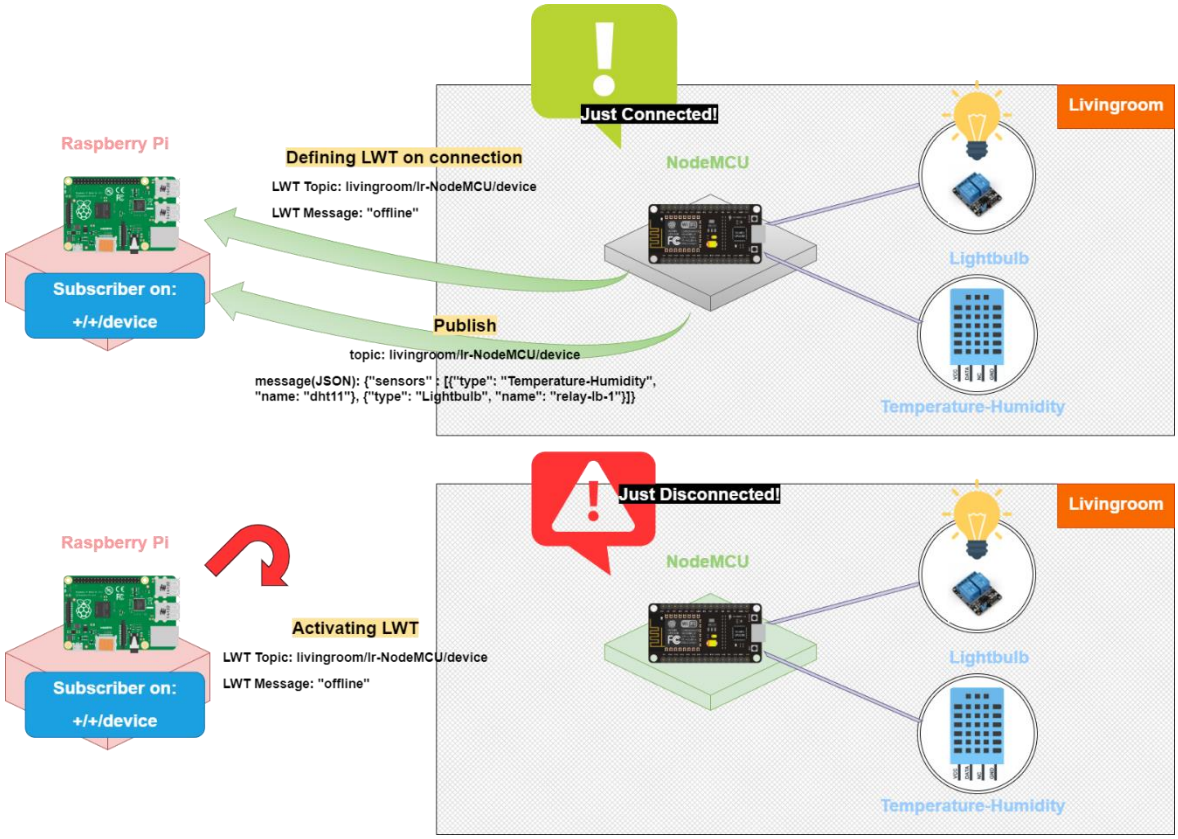
Στο σύστημα οικιακών αυτοματισμών που υλοποιούμε, υπάρχουν δυο είδη topics που αφορούν τον server και τα sensor nodes. Αυτά που αφορούν τη σύνδεση/αποσύνδεση των clients από τον broker και αυτά που σχετίζονται με τη μετάδοση δεδομένων σχετικών με τους αισθητήρες.

Όσον αφορά τη πρώτη κατηγορία, το σκοπό αυτό επιτυγχάνει η έννοια του Last Will Testament (LWT). Στο MQTT, το LWT αφορά ένα topic και ένα message που ορίζει ο client κατά την επίτευξη σύνδεσης του στον broker (βλέπε Σχήμα 6.2, συνάρτηση client.connect()), πληροφορία που ο τελευταίος αποθηκεύει. Στο topic αυτό θα στείλει ο broker το συγκεκριμένο message, μόνο όταν ο client αποσυνδεθεί [23].

Στο σύστημα μας, το LWT χρησιμοποιείται ως εξής:

- Κάθε client έχει ένα LWT Topic, της μορφής «<room-name>/<device-mqtt-client-id>/device», όπου <room-name> το όνομα του δωματίου που βρίσκεται το board, και όπου <device-mqtt-client-id> το μοναδικό id που έχει ως MQTT Client. Το LWT message που στέλνετε μαζί είναι το “offline”.
- Ο broker είναι subscriber στο topic «+/+/device» καλύπτοντας έτσι όλες τις συσκευές που συνδέονται.
- Κατά την σύνδεση τους, οι clients κάνουν publish στο συγκεκριμένο topic πληροφορίες σχετικά με τους αισθητήρες που έχουν καλωδιωμένους, σε μορφή JSON, με το ακόλουθο περιεχόμενο για τη διαχείριση τους από την εφαρμογή του server:
  - {“sensors”: [{“type”: <Sensor1-Type>, “name:”: <Sensor1-Name>}, {“type”: <Sensor2-Type>, “name:”: <Sensor2-Name>, …}]}
- Κατά την αποσύνδεση τους, ο broker, ως subscriber στο LWT Topic κάθε συσκευής, λαμβάνει το LWT message “offline” και έπειτα η node εφαρμογή κάνει τις απαραίτητες ενέργειες που αφορούν αυτή τους την αποσύνδεση.

Στο Σχήμα 6.2 προηγουμένως, φαίνεται αναλυτικά ο τρόπος με τον οποίον ο κώδικας του board, ορίζει ως ένας MQTT client, το LWT, ενώ στο Σχήμα 6.3 απεικονίζεται ένα σχεδιάγραμμα που βοηθάει στη κατανόηση χρήσης το LWT, όπως αυτή παρουσιάστηκε παραπάνω.



Σχήμα 6.3: Απεικόνιση παραδείγματος της MQTT επικοινωνίας του συστήματος κατά τη σύνδεση(πάνω διάγραμμα) ή αποσύνδεση (κάτω διάγραμμα) της συσκευής

6.5 Τα MQTT Topics στη μετάδοση των δεδομένων των αισθητήρων

Έχοντας δυο κατηγορίες αισθητήρων που μπορεί να υπάρχουν πάνω σε ένα sensor node, αυτή των αισθητήρων που συλλέγουν δεδομένα αποκλειστικά (π.χ. αισθητήρας θερμοκρασίας-υγρασίας, αισθητήρας ανίχνευσης κίνησης) και αυτή που περιλαμβάνει τους αισθητήρες που πραγματοποιούν μια κατά-εντολή ενέργεια ή αλλιώς actuators (π.χ. κανάλι ρελέ με τη λάμπα, κανάλι ρελέ με τον θερμοστάτη), χωρίζουμε τα σχετικά topics ως εξής:

- Στους αισθητήρες που συλλέγουν αποκλειστικά δεδομένα, αντιστοιχεί ένα topic που αφορά την πληροφορία που παρέχουν. Συνεπώς, το sensor node στο οποίο

είναι καλωδιωμένος ένας τέτοιος αισθητήρας, με το που συλλέξει τα δεδομένα, τα κάνει publish στο αντίστοιχο topic.

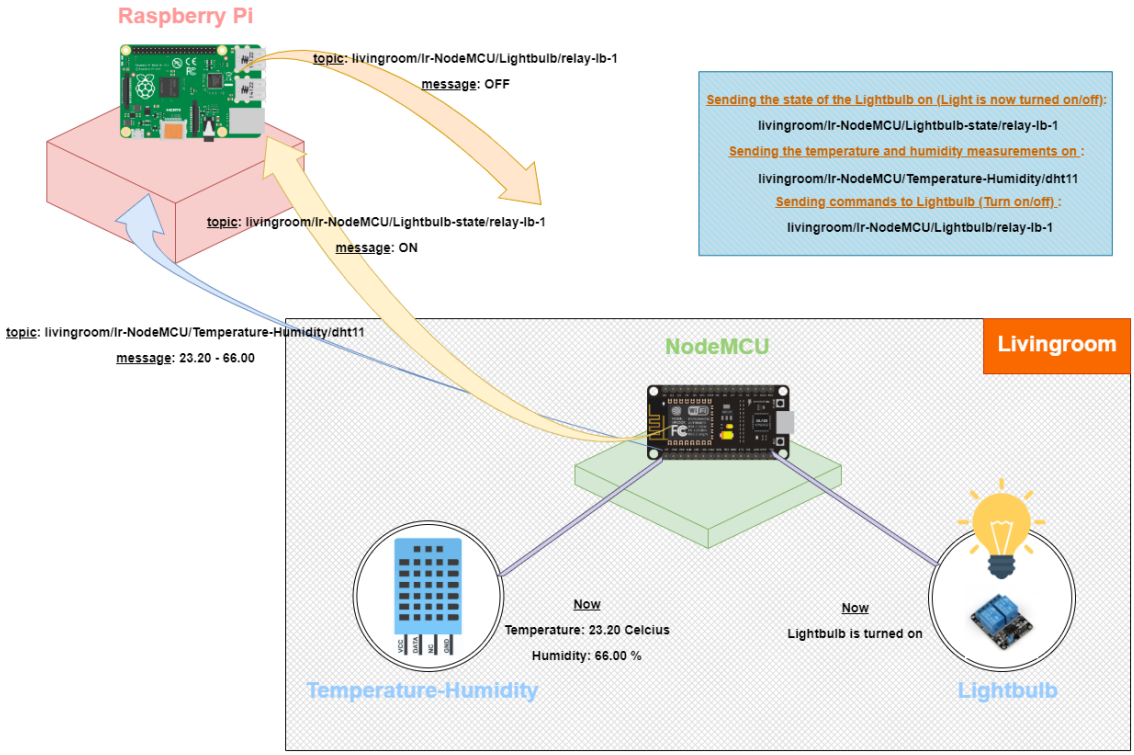
- Στους actuators αντιστοιχούν στον καθένα από δυο topics, ένα για τις εντολές με τις οποίες ο actuator πραγματοποιεί μια ενέργεια και ένα για τη πληροφορία της τωρινής του κατάστασης. Το sensor node στο οποίο ανήκει ο actuator, είναι subscriber στο topic εντολών, ενώ κάνει publish την τωρινή κατάσταση του actuator στο δεύτερο topic.

Επιπλέον, κάθε topic που αφορά έναν αισθητήρα ακολουθεί αυστηρά την ακόλουθη δομή.

- Για τους αισθητήρες που συλλέγουν δεδομένα αποκλειστικά:
  - «<room-name>/<device-mqtt-client-id>/<sensor-type>/<unique-sensor-name>», όπου στέλνονται τα δεδομένα που συλλέγονται από τον αισθητήρα
- Για τους actuators:
  - «<room-name>/<device-mqtt-client-id>/<sensor-type>/<unique-sensor-name>», όπου στέλνονται οι εντολές για τον actuator
  - «<room-name>/<device-mqtt-client-id>/<sensor-type>+"-state”/<unique-sensor-name>», όπου γίνεται publish η κατάσταση του εκείνη τη στιγμή

Στη παραπάνω δομή διευκρινίζεται ότι, όπου <room-name> το όνομα του δωματίου, όπου <device-mqtt-client-id> το client id του microcontroller που λειτουργεί ως MQTT Client, όπου <sensor-type> το είδος του αισθητήρα, και όπου <unique-sensor-name> ένα μοναδικό όνομα που ορίζεται για τον αισθητήρα.

Στο Σχήμα 6.4 φαίνεται ένα παράδειγμα της ροής των δεδομένων μέσα από τα MQTT topics που αναφέρθηκαν, ανάμεσα στο Raspberry Pi και ένα sensor node το οποίο έχει ως microcontroller το NodeMCU και πάνω σε αυτό βρίσκεται ένας αισθητήρας θερμοκρασίας-υγρασίας και ένα ρελέ που οδηγεί μια λάμπα.



Σχήμα 6.4: Απεικόνιση παραδείγματος της μετάδοσης δεδομένων μέσα από τα MQTT Topics, μεταξύ ενός sensor node (MQTT Client) και του Raspberry Pi (MQTT Broker)

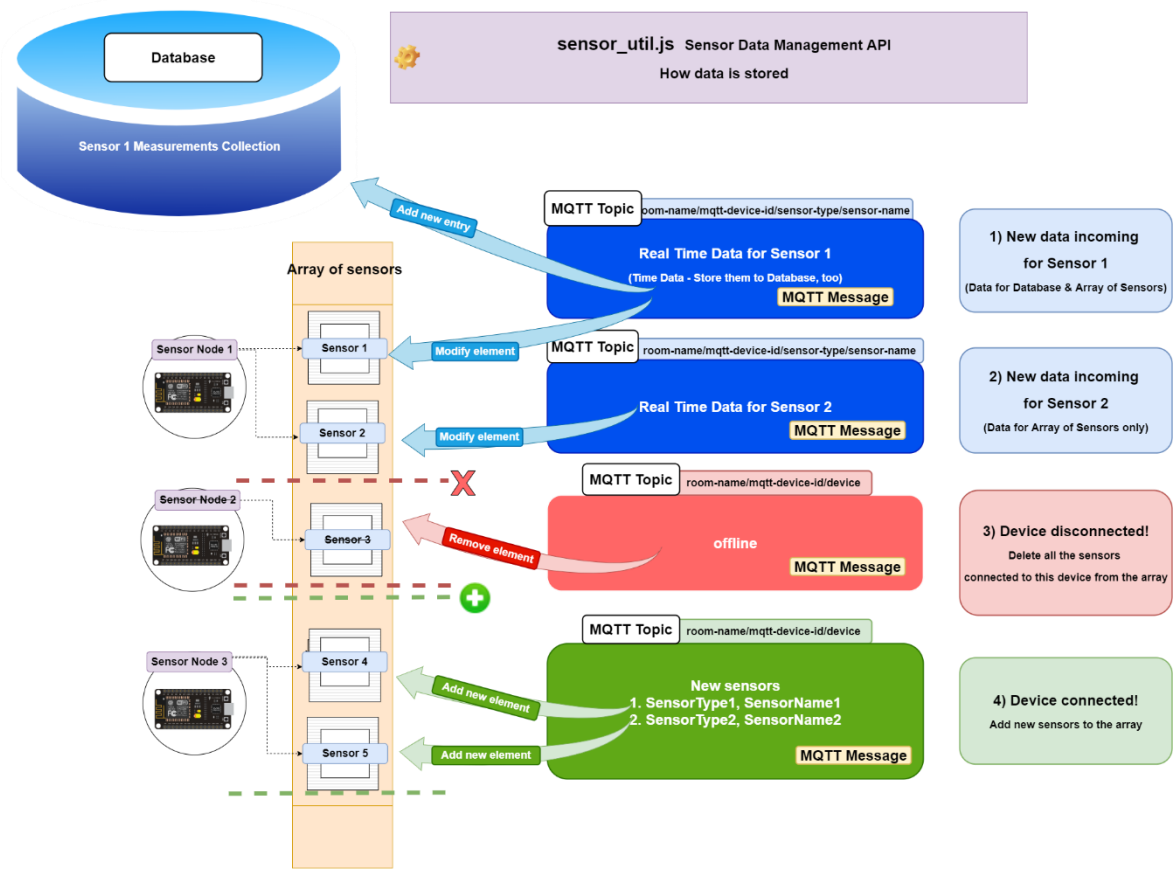
6.6 Επεξεργασία και αποθήκευση των δεδομένων στον server

Η δομή των topics που αναφέρθηκαν στη προηγούμενη ενότητα ακολουθείται αυστηρά για κάθε συσκευή και για κάθε νέο αισθητήρα που προστίθεται, διότι είναι ζωτικής σημασίας για τον server να διαθέτει και άρα να μπορεί να επεξεργαστεί όλες τις απαραίτητες πληροφορίες που αφορούν το σύστημα (π.χ. το όνομα του δωματίου που ανήκουν τα sensor nodes με τους αισθητήρες).

Για την αξιοποίηση των δεδομένων που πλέον μπορούν και καταφθάνουν μέσω των MQTT topics, πρέπει να προστεθεί στον server η ανάλογη λειτουργικότητα που οδηγεί στην αναμενόμενη επεξεργασία και αποθήκευση τους. Συνεπώς, πρέπει να κατασκευαστεί κατάλληλο API διαχείρισης αυτών των δεδομένων.

Τη δυνατότητα αυτή κατέχει το sensor_util.js module. Ο συγκεκριμένος κώδικας υλοποιήθηκε μέσα στην εφαρμογή του server και αποτελεί τον κορμό της επεξεργασίας των δεδομένων που προέρχονται από τα MQTT Topics. Στο διάγραμμα που παρουσιάζεται στο Σχήμα 6.5, φαίνονται οι βασικές λειτουργίες του API και ο τρόπος με τον οποίον

χειρίζεται τα εισερχόμενα δεδομένα. Συγκεκριμένα, στο εσωτερικό του, το module περιέχει έναν καθολικό και δυναμικού μεγέθους πίνακα, ο οποίος κρατά όλους τους αισθητήρες του συστήματος που προέρχονται από τα συνδεδεμένα sensor nodes. Επιπλέον, αναλόγως τον τύπο του αισθητήρα, τα δεδομένα αποθηκεύονται είτε μόνο στον πίνακα των αισθητήρων είτε και στον πίνακα και στη βάση δεδομένων μαζί.



Σχήμα 6.5: Απεικόνιση της λειτουργικότητας του sensor_util.js που περιέχει το API που αναλαμβάνει τη διαχείριση των δεδομένων που έρχονται από τα διάφορα topics.

Στην περίπτωση 1 του διαγράμματος, περιγράφεται η διπλή αυτή αποθήκευση των δεδομένων. Για παράδειγμα, όταν πρόκειται για τον αισθητήρα θερμοκρασίας-υγρασίας DHT11, το API γνωρίζει και αποθηκεύει τις νέες μετρήσεις στο σχετικό πεδίο του αισθητήρα στον πίνακα, αντικαθιστώντας έτσι τη παλιά μέτρηση. Ταυτόχρονα, προσθέτει στη βάση δεδομένων τη νέα πληροφορία, σε ένα MongoDB collection που φέρει το όνομα του αισθητήρα και περιέχει όλες τις μετρήσεις αντιστοιχισμένες σε «buckets» που αναλογούν σε κάθε μισάωρο της μέρας (π.χ. ένα bucket μπορεί να περιέχει τις μετρήσεις

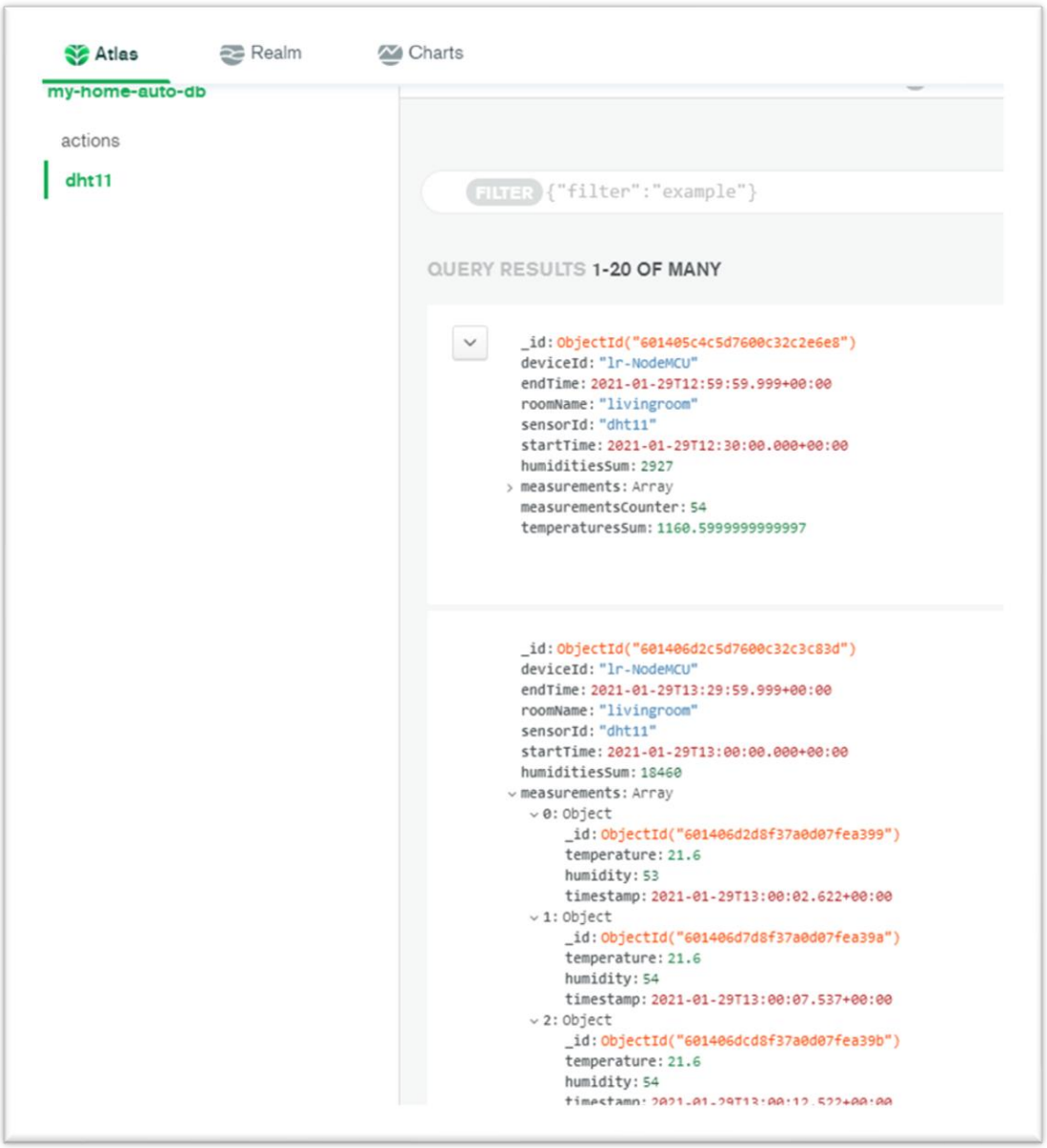
που έχουν γίνει από 8:30 μ.μ. μέχρι 9:00 μ.μ., το επόμενο από αυτό, τις μετρήσεις από 9 μ.μ. μέχρι 9:30 μ.μ. και ούτω καθεξής).

Στην περίπτωση 2 του διαγράμματος, περιγράφεται η απλή αποθήκευση της πληροφορίας ενός αισθητήρα. Αναλυτικά, όταν έρχονται σε πραγματικό χρόνο νέα δεδομένα από έναν αισθητήρα για τον οποίον δε μας ενδιαφέρει η καταγραφή της πληροφορίας τους στον χρόνο, αυτή αποθηκεύεται αποκλειστικά στον πίνακα, αντικαθιστώντας από τη σχετική θέση τη παλιά πληροφορία. Ο αισθητήρας BH1750FVI που μετρά τη φωτεινότητα είναι ένα σχετικό παράδειγμα.

Στην περίπτωση 3 και 4 έχουμε τη διαχείριση της σύνδεσης ή αποσύνδεσης των συσκευών. Όταν ένα sensor node αποσυνδέεται και έρχεται στον broker το μήνυμα «offline» από το topic «+/+/device», το API προσπελαύνει τον πίνακα και διαγράφει όλους τους αισθητήρες που του αναλογούν (περίπτωση 3). Αντίθετα, όταν στο ίδιο topic έρχεται το μήνυμα που περιέχει σε μορφή JSON τους τύπους και τα ονόματα αισθητήρων, αυτό σημαίνει ότι ένα νέο sensor node συνδέθηκε στο σύστημα. Επομένως, το API αναλαμβάνει να προσθέσει στον πίνακα τους νέους αισθητήρες (περίπτωση 4).

Στο Σχήμα 6.6 μπορεί να φανεί ένα πραγματικό στιγμιότυπο των στοιχείων του πίνακα των αισθητήρων μέσα από τη χρησιμοποίηση του sensor_util API που αναφέρθηκε. Παράλληλα, στο Σχήμα 6.7, παρουσιάζεται ο τρόπος αποθήκευσης των δεδομένων μέσα από το ίδιο API, για τον αισθητήρα θερμοκρασίας-υγρασίας, στη βάση δεδομένων MongoDB. Στη βάση, αποθηκεύονται σε μισάωρα “buckets” της μέρας όλες οι μετρήσεις θερμοκρασίας και υγρασίας που έγιναν σε κάθε αντίστοιχο μισάωρο μαζί με το άθροισμα των μετρήσεων και τον συνολικό αριθμό τους.

Σχήμα 6.6: Απεικόνιση των δεδομένων σε μορφή JSON των αισθητήρων ανίχνευσης κίνησης και ανίχνευσης έντασης φωτεινότητας, σε ένα στιγμιότυπο του πίνακα της εφαρμογής, όπου και αποθηκεύονται



Σχήμα 6.7: Απεικόνιση των αποθηκευμένων δεδομένων του αισθητήρα θερμοκρασίας-υγρασίας DHT11 στη βάση, μέσα από το user interface του MongoDB Atlas

ΚΕΦΑΛΑΙΟ 7

Ο WEB CLIENT ΚΑΙ ΟΙ ΒΑΣΙΚΕΣ ΤΟΥ ΛΕΙΤΟΥΡΓΙΕΣ

7.1 Εισαγωγή

Για να είναι ολοκληρωμένο ένα σύστημα οικιακών αυτοματισμών, απαιτείται η δημιουργία αλληλεπίδρασης με τα δεδομένα που συλλέγονται. Ο χρήστης του συστήματος θα πρέπει να μπορεί μέσα από ένα φιλικό προς τη χρήση του γραφικό περιβάλλον, να παρακολουθεί τα δεδομένα των διαφόρων αισθητήρων που έχει στο σπίτι του, να δίνει εντολές σε αυτά και το βασικότερο, να θέτει ενέργειες που πραγματοποιούνται αυτόματα.

Συνεπώς, η κατασκευή μιας client εφαρμογής για τις καθημερινές συσκευές που ήδη χρησιμοποιεί, επιτυγχάνει ακριβώς αυτή την ανάγκη. Στο σύστημα που περιγράφεται, υλοποιούμε τόσο έναν Web client συμβατό για φορητές και μη συσκευές (υπολογιστής, laptop, tablet, smartphone) και στη συνέχεια μετατρέπουμε τον ίδιο κώδικα, σε native Android εφαρμογή (για την αξιοποίηση της τοποθεσίας του χρήστη όπως θα δούμε στη συνέχεια).

Συνοπτικά, το User Interface του συστήματος θέλουμε να προσφέρει τις εξής δυνατότητες στο σύνολο του:

- Προβολή της πληροφορίας των συνδεμένων αισθητήρων με ευανάγνωστο τρόπο και σε πραγματικό χρόνο
- Πραγματοποίηση ενεργειών μέσα από το γραφικό περιβάλλον (π.χ. ύπαρξη διακόπτη για το άνοιγμα/κλείσιμο της λάμπας)
- Φιλτράρισμα στη προβολή των συνδεμένων αισθητήρων με βάση το δωμάτιο που βρίσκονται ή το είδος τους
- Ορισμός τριών ειδών ενεργειών – αυτοματισμών προς τους αισθητήρες που δέχονται εντολές. Αυτές είναι:
    - Δρομολόγηση ενέργειας σε μια συγκεκριμένη ημερομηνία, με επιλογή επανάληψης
    - Δρομολόγηση ενέργειας με βάση την μέτρηση ή κατάσταση ενός άλλου αισθητήρα

- Δρομολόγηση ενέργειας με βάση την απόσταση του χρήστη από το σπίτι του (για την Android εφαρμογή αποκλειστικά)

- Εμφάνιση ειδοποιήσεων για τη σύνδεση / αποσύνδεση κάποιου sensor node αλλά και για τη στιγμή που ενεργοποιείται μια αυτόματη ενέργεια που προηγουμένως έχει οριστεί από τον χρήστη
- Προστασία της εφαρμογής με ορισμό username και password και άρα δυνατότητες Login / Logout

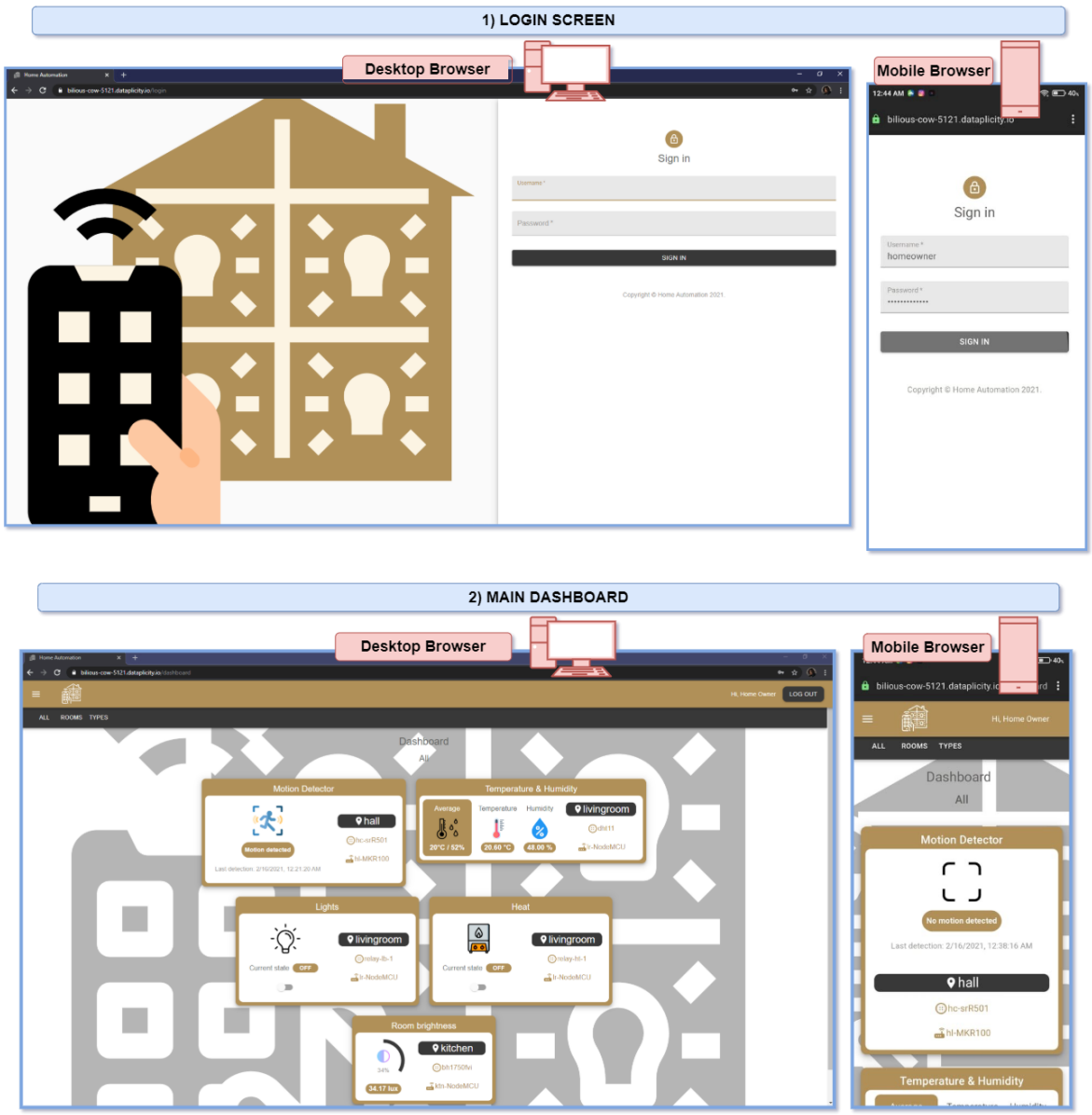
7.2 Κύρια απεικόνιση του γραφικού περιβάλλοντος

Όπως αναφέρθηκε στο Κεφάλαιο 4 και κατά τον προσδιορισμό των software εργαλείων που χρησιμοποιούνται στην ανάπτυξη του συστήματος, η client εφαρμογή υλοποιείται μέσω του React [24]. Επιπλέον, με σκοπό να προσδώσουμε στο γραφικό περιβάλλον έναν βαθμό κομψότητας και ταυτόχρονα να διευκολύνουμε την υλοποίηση του, χρησιμοποιείται μαζί με το React, ακόμη μια JavaScript βιβλιοθήκη, το Material UI [25]. Η βιβλιοθήκη αυτή παρέχει μια πληθώρα στοιχείων όπως Buttons, Switches και App Bars, τα οποία μπορούν να χρησιμοποιηθούν κατά την ανάπτυξη της εφαρμογής.

Πριν αναλυθεί ο τρόπος επικοινωνίας του client με τον server για τη πρόσβαση στα δεδομένα του συστήματος, είναι σημαντικό σε πρώτο στάδιο παρουσιαστούν οι λειτουργίες του στη πράξη, όντας ολοκληρωμένες. Επιπρόσθετα, είναι συνετό να δείξουμε τη τελική μορφή του γραφικού περιβάλλοντος που βλέπει ο χρήστης όταν ανοίγει τον client στον Web Browser του υπολογιστή ή του κινητού του, έτσι όπως αυτή υλοποιήθηκε μέσα από τις βιβλιοθήκες της React και του Material UI.

Οθόνες:

Ο client έχει στο σύνολο του δύο οθόνες, αυτή της εισόδου στοιχείων για ταυτοποίηση (Login Screen) και την κύρια οθόνη της εφαρμογής το Dashboard. Στο Σχήμα 7.1, παρουσιάζονται οι δύο οθόνες και η εμφάνιση τους είτε σε Desktop είτε σε Smartphone Web Browser. Στην Login οθόνη, ο χρήστης πληκτρολογεί το username και το password, και αν αυτά είναι σωστά, τότε μεταφέρεται στο Dashboard. Στην οθόνη του Dashboard, στη συνέχεια, έχει πρόσβαση σε όλους τους αισθητήρες της οικίας, αλλά και την επιλογή μέσα από ένα LOG OUT Button να ξανά-επιστρέψει στην οθόνη Login.



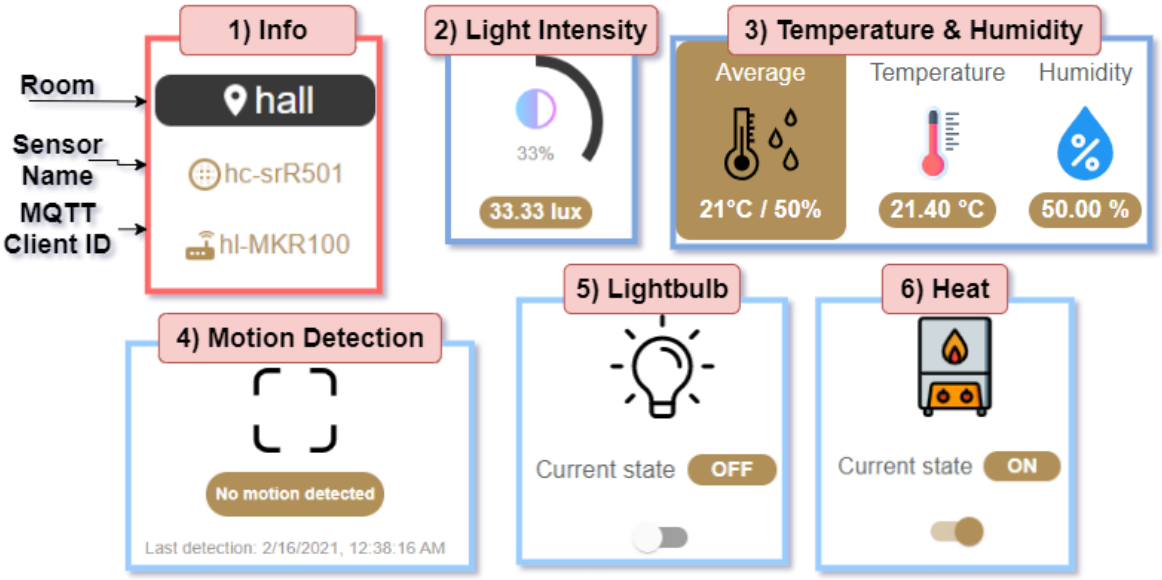
Σχήμα 7.1: Απεικόνιση των οθονών του γραφικού περιβάλλοντος, όπως φαίνονται σε Desktop και Mobile Web Browsers. 1) Login Screen, 2) Main Dashboard

Δεδομένα αισθητήρων:

Στο Σχήμα 7.2 παρουσιάζεται η απεικόνιση των συνδεμένων αισθητήρων και των δεδομένων τους, όπως εμφανίζονται στο Dashboard του γραφικού περιβάλλοντος. Λαμβάνοντας υπόψη την αρίθμηση στο Σχήμα, έχουμε:

1) Η πληροφορία για το δωμάτιο που βρίσκεται ο αισθητήρας, το όνομα του αισθητήρα, και το MQTT Client Id του sensor node στο οποίο ανήκει. Εμφανίζεται σε όλες τις κάρτες των αισθητήρων.

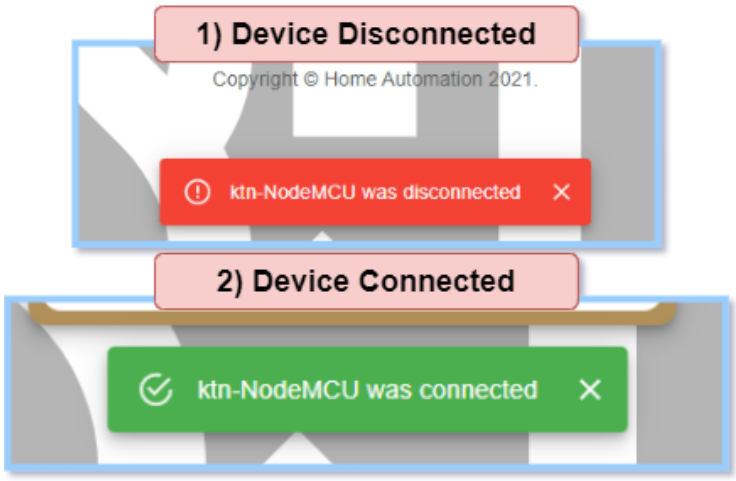
2) Η απεικόνιση της φωτεινότητας στο δωμάτιο εκείνη τη στιγμή από τον αισθητήρα μέτρησης της έντασης φωτεινότητας. Η ένταση του φωτεινότητας μετριέται σε lux και απεικονίζεται η μέτρηση της στη κάρτα, όπως επίσης και το σχετικό ποσοστό της.
3) Η απεικόνιση των μετρήσεων της θερμοκρασίας και της υγρασίας εκείνη τη στιγμή. Μαζί με τις μετρήσεις σε πραγματικό χρόνο, η κάρτα απεικονίζει ακόμη και τη μέση τιμή τους για την τελευταία ώρα.
4) Η απεικόνιση της κάρτας του αισθητήρα ανίχνευσης κίνησης με το σχετικό μήνυμα. Σε περίπτωση εντοπισμού κίνησης, το εικονίδιο μετατρέπεται στο αντίστοιχο του γεγονότος της κίνησης. Επιπλέον, στη κάρτα παρουσιάζεται και η ακριβής ημερομηνία και ώρα του τελευταίου εντοπισμού.
5) Η απεικόνιση της κάρτας της λάμπας συνδεμένης στο ρελέ. Το εικονίδιο της κάρτας αλλάζει ανάλογα με το άνοιγμα ή το κλείσιμο της, ενώ το σχετικό μήνυμα υποδεικνύει τη κατάσταση της γραπτώς (ON/OFF). Μέσα από το σχετικό διακόπτη μπορεί ο χρήστης να αλλάξει τη κατάσταση της.
6) Η απεικόνιση της κάρτας του θερμοστάτη συνδεμένου στο ρελέ. Η κάρτα είναι πανομοιότυπη με αυτή της λάμπας αλλά με διαφορετικό εικονίδιο που υποδεικνύει ότι αφορά τη θέρμανση.



Σχήμα 7.2: Απεικόνιση της πληροφορίας που παρουσιάζει κάθε συνδεμένος αισθητήρας στο γραφικό περιβάλλον

Ειδοποιήσεις:

Όταν ένα sensor node συνδέεται στο ρεύμα και κατόπιν εισέρχεται στο σύστημα, τότε στον client εμφανίζεται η σχετική ειδοποίηση και αυτόματα όλοι αισθητήρες που είναι καλωδιωμένοι στο sensor node, εμφανίζονται στο Dashboard. Για την αποσύνδεση ενός sensor node από το σύστημα, αναλόγως ο client εμφανίζει μια ειδοποίηση σχετικά με το γεγονός και αυτομάτως όλοι αισθητήρες που ανήκαν στο sensor node εξαφανίζονται αυτομάτως από το Dashboard. Στο Σχήμα 7.3 διακρίνονται αυτές οι 2 περιπτώσεις.



Σχήμα 7.3: Απεικόνιση των ειδοποιήσεων για τη 1) σύνδεση ή 2) αποσύνδεση ενός sensor node

Περιήγηση:

Στο Σχήμα 7.4 παρουσιάζεται ο τρόπος που γίνεται η περιήγηση στο γραφικό περιβάλλον της εφαρμογής. Λαμβάνοντας υπόψη την αρίθμηση στο Σχήμα, έχουμε:

1) Menu Bar. Μέσω του Menu Bar είναι εφικτό το φιλτράρισμα στη προβολή των εικονιζόμενων συνδεμένων αισθητήρων. Πατώντας το ALL εμφανίζονται όλοι, ενώ με το ROOMS και το TYPES εμφανίζονται τα modals που περιγράφονται στο 4) και στο 5) αντίστοιχα.
2) Το LOG OUT κουμπί που επιτρέπει την επιστροφή στο Login Screen.
3) Ο Drawer του γραφικού περιβάλλοντος. Εμφανίζεται αν γίνει click στο πάνω αριστερά Button με τις τρεις γραμμές στην οθόνη του Dashboard και αποκαλύπτει τις ίδιες

επιλογές με το Menu Bar μαζί με δυο επιπλέον, μια για τα Actions (οι αυτοματισμοί) και μια για Log Out και επιστροφή στην Login οθόνη.

4) Το modal για τα δωμάτια που δίνει την επιλογή στον χρήστη να επιλέξει τη προβολή των αισθητήρων μόνο για ένα συγκεκριμένο δωμάτιο.
5) Το modal για τους τύπους των αισθητήρων που δίνει την επιλογή στον χρήστη να επιλέξει τη προβολή των αισθητήρων που έχουν τον ίδιο τύπο (π.χ. τις θερμοκρασίες-υγρασίες σε όλα τα δωμάτια).

Σχήμα 7.4: Απεικόνιση όλων των τρόπων περιήγησης στην εφαρμογή

ΚΕΦΑΛΑΙΟ 8

ΟΡΙΣΜΟΣ ΑΥΤΟΜΑΤΙΣΜΩΝ ΣΤΟΝ WEB CLIENT

8.1 Εισαγωγή

Στο προηγούμενο Κεφάλαιο αναφέρθηκε, αλλά σκοπίμως δεν αναλύθηκε ώστε να παρουσιαστεί εδώ, ο ορισμός αυτοματισμών που μπορεί να θέσει ο χρήστης στο σύστημα μέσα από το γραφικό περιβάλλον. Όπως τονίστηκε, ο χρήστης πρέπει να μπορεί να θέσει τρία είδη αυτοματισμών στο σύστημα και αυτά είναι:

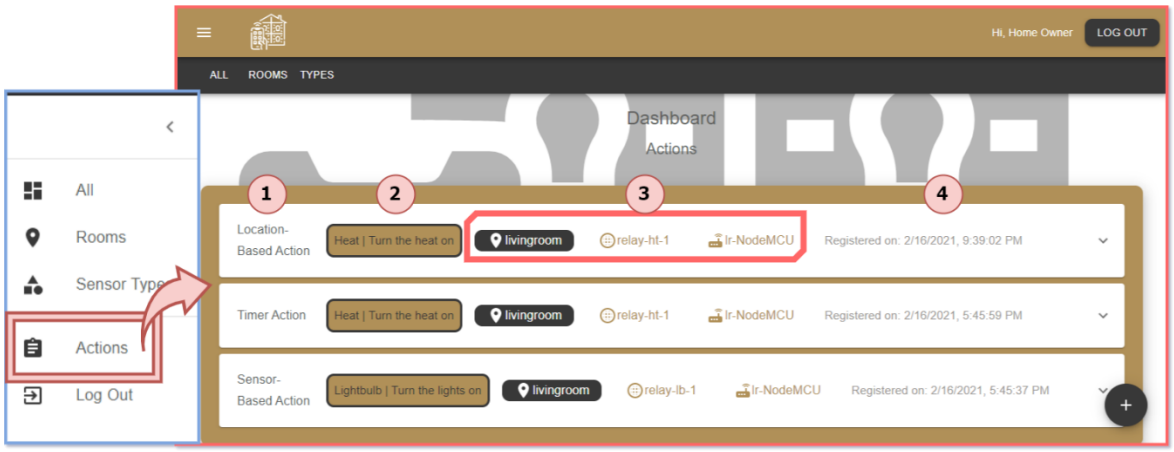
- Δρομολόγηση ενέργειας σε μια συγκεκριμένη ημερομηνία, με επιλογή επανάληψης
- Δρομολόγηση ενέργειας με βάση την μέτρηση ή κατάσταση ενός άλλου αισθητήρα
- Δρομολόγηση ενέργειας με βάση την απόσταση του χρήστη από το σπίτι του

Το Κεφάλαιο 8 αναλύει τα δύο πρώτα είδη αυτοματισμών που αφορούν τον web client, ενώ στο Κεφάλαιο 9 θα επισημανθεί η τρίτη λειτουργία, η οποία αφορά την τοποθεσία, και είναι διαθέσιμη μόνο στην native εφαρμογή για Android τηλέφωνα.

8.2 Ορισμός αυτοματισμών μέσα από το γραφικό περιβάλλον

Ο χρήστης θέτει τους αυτοματισμούς ή Actions (το όνομα τους στην αργκό της εφαρμογής που υλοποιούμε) μέσα από το γραφικό περιβάλλον, ενώ παράλληλα μπορεί μέσα από αυτό να δει μια λίστα από αυτούς έχει ήδη θέσει. Στο Σχήμα 8.1, φαίνεται η περιήγηση στη σελίδα των Actions που γίνεται μέσα από τον Drawer του client. Η σελίδα των Actions παρουσιάζει όσα Actions έχουν τεθεί, από το νεότερο προς το παλαιότερο χρονικά, ενώ για καθένα από αυτά φέρει τις εξής κοινές πληροφορίες:

1) Το είδος του αυτοματισμού (Timer Action, Location-Based Action, Sensor-Based Action)
2) Ο τύπος του αισθητήρα - actuator που στοχεύει ο αυτοματισμός και η εντολή υπό συνθήκη που θα εκτελέσει.
3) Τα στοιχεία του αισθητήρα – actuator (το δωμάτιο που βρίσκεται, το όνομα του και το MQTT Client Id του sensor node στο οποίο ανήκει)
4) Η ημερομηνία που ορίστηκε ο αυτοματισμός από τον χρήστη



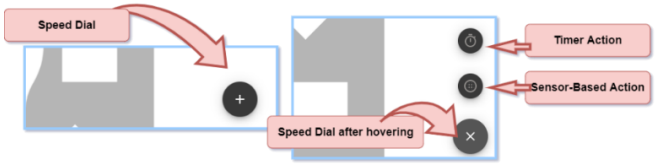
Σχήμα 8.1: Απεικόνιση της σελίδας των αυτοματισμών Actions όπως αυτή φαίνεται στο γραφικό περιβάλλον της εφαρμογής

Κάθε αυτοματισμός που εμφανίζεται στη σελίδα των Actions του γραφικού περιβάλλοντος έχει τη μορφή Ακορντεόν. Αυτό σημαίνει ότι με ένα click πάνω στην επιφάνεια κάποιου Action αποκαλύπτονται επιπρόσθετες πληροφορίες, οι οποίες διαφέρουν ανάλογα με τον τύπο του Action. Στο Σχήμα 8.2 παρουσιάζονται αυτές οι πληροφορίες και συγκεκριμένα:

1) Για τα Timer Actions, τους αυτοματισμούς δηλαδή που τίθενται να πραγματοποιηθούν σε μια συγκεκριμένη ημερομηνία και ώρα, με ή χωρίς επανάληψη ανά συγκεκριμένα χρονικά διαστήματα, παρουσιάζεται αριστερά η ημερομηνία εκτέλεσης τους και δεξιά το διάστημα που μεσολαβεί για την επανάληψη τους. Εφόσον δεν υπάρχει επανάληψη, το τελευταίο πεδίο είναι None.
2) Για τα Sensor-Based Actions, τους αυτοματισμούς δηλαδή που βασίζονται στη μέτρηση κάποιου αισθητήρα για να ενεργοποιηθούν, παρουσιάζονται ο τύπος του αισθητήρα μαζί με τις πληροφορίες του, καθώς επίσης και ένα μήνυμα που εξηγεί ποια πρέπει να είναι η μέτρηση του για να εκτελεστεί ο αυτοματισμός.
3) Για τα Location-Based Actions (τα οποία έχουν τεθεί από την εφαρμογή Android του συστήματος), παρουσιάζεται η μέγιστη ακτίνα σε μέτρα γύρω από την οικία του χρήστη, για την οποία εφόσον ο χρήστης βρεθεί εντός της, ενεργοποιείται το αντίστοιχο Action.
4) Το κουμπί διαγραφής που είναι κοινό για όλα τα είδη αυτοματισμών και μέσω αυτού ο χρήσης μπορεί να διαγράψει το Action για το οποίο έκανε click, αφού πρώτα δει το μήνυμα με τη σχετική προειδοποίηση της διαγραφής.

Σχήμα 8.2: Απεικόνιση των πληροφοριών για κάθε διαφορετικό είδος Action, έτσι όπως αυτά προβάλλονται στο γραφικό περιβάλλον

Αφού αναλύθηκε ο τρόπος με τον οποίον εμφανίζονται τα κατοχυρωμένα Actions, σειρά έχει η επισήμανση του τρόπου με τον οποίο αυτά ορίζονται. Στο Σχήμα 8.3, παρουσιάζεται το πως μπορεί να επιτευχθεί ο ορισμός ενός αυτοματισμού. Ο χρήστης στην σελίδα των Actions βρίσκει διαθέσιμο ένα Speed Dial (βλέπε κάτω δεξιά, Σχήμα 8.1). Το Speed Dial στο γραφικό περιβάλλον, εμφανίζει πρόσθετες επιλογές που μπορούν να προσπελαστούν, εφόσον ο χρήστης κάνει Hover πάνω του (μετακινήσει το βέλος του ποντικιού πάνω του, χωρίς να κάνει click). Οι επιλογές που εμφανίζονται είναι δύο, μια που αντιστοιχεί στον ορισμό ενός Timer Action και μια που αφορά τον ορισμό Sensor-Based Action. Το click πάνω σε αυτά εμφανίζει ένα σχετικό modal παράθυρο, διαφορετικό για τα δυο είδη των αυτοματισμών.



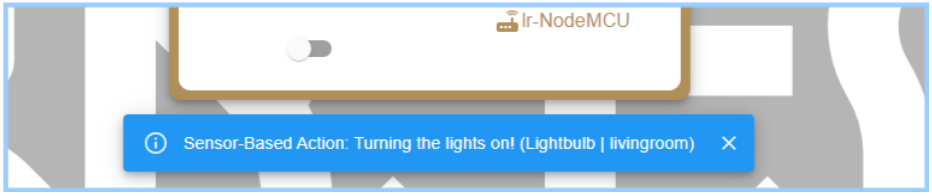
Σχήμα 8.3: Απεικόνιση του Speed Dial και των επιλογών του στην σελίδα Actions της Web Client εφαρμογής

Στο Σχήμα 8.4 φαίνονται τα modals των δυο περιπτώσεων. Αναλυτικά:

1) Το modal για το Timer Action ζητά από τον χρήστη μέσα από το σχετικό μενού να επιλέξει τον αισθητήρα για τον οποίο θέλει να εκτελεστεί κάποια εντολή και εφόσον επιλεχτεί, στη συνέχεια ένα menu εμφανίζεται με τις σχετικές επιλογές εντολών για το συγκεκριμένο αισθητήρα - actuator. Στα αμέσως επόμενα μενού ο χρήστης επιλέγει την ημερομηνία και την ώρα στην οποία θα εκτελεστεί η συγκεκριμένη εντολή, ενώ προαιρετικά έχει μέσα από τα τελευταία στη σειρά πλαίσια(ένα μενού και μια φόρμα συμπλήρωσης) την επιλογή να θέσει επανάληψη στη διαδικασία. Μπορεί να θέσει το είδος του χρονικού διαστήματος (Λεπτά, Ώρες, Μέρες) και να πληκτρολογήσει τον αριθμό του χρονικού διαστήματος (π.χ. Κάθε 2 ώρες, Κάθε 7 μέρες, Κάθε 7 λεπτά και ούτω καθεξής).
2) Το modal για το Sensor-Based Action έχει πολλά κοινά στοιχεία με αυτό του Timer Action. Στο πλαίσιο Trigger ο χρήστης επιλέγει τον αισθητήρα – actuator και την επιθυμητή εντολή προς εκτέλεση. Από την άλλη πλευρά, στο πλαίσιο When επιλέγει τον δεύτερο αισθητήρα, το είδος της μέτρησης και τη σχετική συνθήκη που πρέπει να καλυφθεί για να εκτελεστεί ο αυτοματισμός. Αν ο αισθητήρας δίνει ποσοτικές μετρήσεις (π.χ. ένταση φωτεινότητας), τότε η σχετική συνθήκη αποτελείται από ένα μενού με τις επιλογές Below, Over, Equal To και μια φόρμα συμπλήρωσης του ποσού της μέτρησης. Αν ο αισθητήρας ανήκει σε είδος αισθητήρων που δεν επιστρέφει ποσότητες αλλά συγκεκριμένες επιλογές (π.χ. ανιχνευτής κίνησης με επιλογές Motion Detected, No Motion Detected), τότε αντί για τα παραπάνω σχετικά με τη ποσότητα μενού, εμφανίζεται ένα μοναδικό μενού με αυτές τις τιμές προς επιλογή.
3) Και για τα δύο modals, εφόσον ο χρήστης επιλέξει τις επιλογές που επιθυμεί (εκτός από τα μενού της επανάληψης που είναι προαιρετικά), με το πάτημα του κουμπιού ADD ACTION στέλνεται το σχετικό request στον server. Αν έχουν συμπληρωθεί όλα τα απαραίτητα πεδία, επιστρέφεται θετική απάντηση (εμφανίζεται και σχετικό μήνυμα) από τον server για τη κατοχύρωση του Action, ενώ σε αντίθετη περίπτωση, εμφανίζεται μήνυμα που εξηγεί το σχετικό λάθος.

Σχήμα 8.4: Απεικόνιση των modals για τον ορισμό είτε ενός Timer Action 1) είτε ενός Sensor-Based Action 2), καθώς και της ειδοποίησης 3) για την επιτυχία ή την αποτυχία της εισαγωγής

Όλα τα παραπάνω αφορούν τον ορισμό αυτοματισμών μέσω του γραφικού περιβάλλοντος και η εκτέλεση τους γίνεται αυτόματα από το σύστημα, εφόσον εκπληρωθούν οι συνθήκες που έθεσε ο χρήστης. Για την απόδειξη της εκτέλεσης τους σε πραγματικό χρόνο μέσα από το γραφικό περιβάλλον, προστίθεται η δυνατότητα ειδοποίησης μέσα από σχετικό μήνυμα που αναφέρει το είδος του αυτοματισμού που μόλις ενεργοποιήθηκε, την εντολή που εκτελείται, το είδος του αισθητήρα και το δωμάτιο του σπιτιού (βλέπε Σχήμα 8.5).



Σχήμα 8.5: Απεικόνιση της εμφάνισης ειδοποίησης στο γραφικό περιβάλλον κατά την ενεργοποίηση ενός αυτοματισμού

ΚΕΦΑΛΑΙΟ 9

ANDROID ΕΦΑΡΜΟΓΗ ΚΑΙ ΑΥΤΟΜΑΤΙΣΜΟΙ ΜΕ ΒΑΣΗ ΤΗΝ ΤΟΠΟΘΕΣΙΑ

9.1 Εισαγωγή

Στο προηγούμενα Κεφάλαια, παρουσιάστηκε η κατασκευή της client εφαρμογής του συστήματος μέσω του React JavaScript Framework, για την οποία δείχτηκε ότι είναι συμβατή τόσο για desktop όσο και για mobile web browsers. Επομένως, μια native εφαρμογή Android που μπορεί να εγκατασταθεί σε Android τηλέφωνα θα ήταν αχρείαστη αν δε προσέφερε κάτι παραπάνω από τον αντίστοιχο web client του συστήματος. Συνεπώς, η πρόσθετη λειτουργία που είναι επιθυμητό να προσφέρει μια Android εφαρμογή στο σύστημα είναι ο ορισμός αυτοματισμών με βάση την τοποθεσία.

Σε αντίθεση με τον client στον browser, μια εφαρμογή Android, εφόσον υλοποιηθεί κατά αυτόν τον τρόπο, μπορεί να εντοπίζει την τοποθεσία του χρήστη είτε όταν αυτός έχει ανοιχτή την εφαρμογή στο προσκήνιο, είτε την έχει αφήσει ανοιχτή στο παρασκήνιο χρησιμοποιώντας μια άλλη. Ακόμη, ο εντοπισμός της τοποθεσίας μπορεί να γίνεται και όταν το κινητό βρίσκεται κλειδωμένο με κλειστή την οθόνη.

Για την κατασκευή της υπάρχει μια πληθώρα εργαλείων και γλωσσών προγραμματισμού που μπορούν να χρησιμοποιηθούν, όπως η Java, η Kotlin ή ακόμα και η React Native που αποτελεί το αντίστοιχο React Framework για mobile εφαρμογές. Παρόλα αυτά, η μια και μοναδική πρόσθετη λειτουργία που θέλουμε να προσθέσουμε στην εφαρμογή σε σχέση με τον Web Client, δεν δικαιολογεί τη γραφή κώδικα από την αρχή. Τη λύση στο πρόβλημα αυτό έρχεται να δώσει το εργαλείο Apache Cordova.

Το Apache Cordova, μπορεί να μετατρέψει μια υπάρχουσα web εφαρμογή που αποτελείται από το τρίπτυχο HTML5/CSS3/JavaScript σε μια native εφαρμογή κινητών τηλεφώνων τοποθετώντας έναν wrapper πάνω από αυτές τις τεχνολογίες, ο οποίος στοχεύει μια επιλεγμένη πλατφόρμα, όπως αυτή του Android. Επιπρόσθετα, μέσω διαφόρων APIs που χρησιμοποίει, επιτρέπει την αξιοποίηση των λειτουργιών του τηλεφώνου, όπως το δίκτυο, τη κάμερα και τη τοποθεσία, ενώ η λειτουργικότητα μιας τέτοιας εφαρμογής μπορεί να επεκταθεί με τη χρήση διαφόρων plugins που προσφέρει [26].

9.2 Υλοποίηση της εφαρμογής και πρόσβαση στην τοποθεσία του κινητού

Αφού εγκαταστήσουμε το Cordova, πρέπει στη συνέχεια να εισάγουμε τον υπάρχοντα κώδικα του web client σε αυτό. Επειδή ο κώδικας είναι γραμμένος με τη βοήθεια της React βιβλιοθήκης, η διαδικασία της μετατροπής απαιτεί τον ορισμό συγκεκριμένων ρυθμίσεων για επιτευχθεί ομαλή μετατροπή από React σε Cordova. Σύμφωνα με τον οδηγό [27] τοποθετούμε τις απαραίτητες ρυθμίσεις στα κατάλληλα αρχεία και εισάγουμε τον κώδικα σε μια νεοσύστατη Cordova εφαρμογή, η οποία βρίσκεται πλέον στο ίδιο σημείο ανάπτυξης με τον web client.

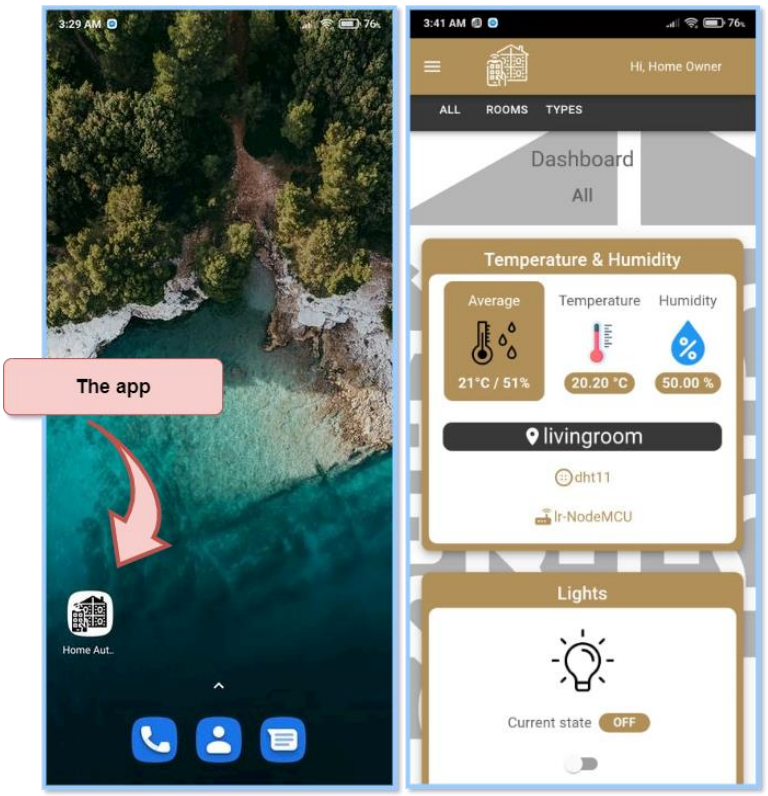
Για να προσθέσουμε τη δυνατότητα πρόσβασης της εφαρμογής στη τοποθεσία του κινητού τηλεφώνου, είναι απαραίτητη η εγκατάσταση ενός Cordova plugin με όνομα cordova-plugin-background-geolocation [28]. Το plugin αυτό προσφέρει ένα API μέσω του οποίου μπορούμε να ορίσουμε το διάβασμα της τοποθεσίας από την εφαρμογή, όταν αυτή βρίσκεται στο προσκήνιο ή το παρασκήνιο ή όταν το κινητό είναι κλειδωμένο. Επιπρόσθετα, μέσα από το API, μπορούμε να ορίσουμε διάφορες ρυθμίσεις, όπως την ακρίβεια στην εύρεση της τοποθεσίας, το χρόνο ανάμεσα στα διαβάσματα που επιχειρεί και το URL στο οποίο μπορεί να στέλνεται η πληροφορία της τοποθεσίας μόλις γίνει η μέτρηση. Η τελευταία ρύθμιση είναι η κυριότερη, καθώς για κάθε διάβασμα της τοποθεσίας, ορίζεται αυτό να στέλνεται στο URL που αντιστοιχεί στον server του συστήματος.

Σημαντική σημείωση που πρέπει να τονιστεί, αποτελεί το γεγονός ότι για Android συσκευές που φέρουν την έκδοση Android 10 και πάνω, θα πρέπει η εφαρμογή, μέσα από τον κώδικα της, να ενημερώνει υποχρεωτικά τον χρήστη ότι επιθυμεί να εντοπίζει την τοποθεσία του στο παρασκήνιο και να ζητά την άδεια του. Επομένως η εφαρμογή που υλοποιείται πρέπει να καλύπτει αυτό το γεγονός ενεργοποιώντας ή κρατώντας απενεργοποιημένες τις υπηρεσίες τοποθεσίας που προσφέρει η εφαρμογή [29].

Αφού λοιπόν υλοποιήσουμε τη χρήση του plugin στον κώδικα και αυτός είναι πλέον σε θέση να διαβάζει την τοποθεσία κατάλληλα, συνδέουμε το κινητό με USB στον υπολογιστή που τρέχει το εργαλείο Cordova και εγκαθιστούμε μέσω αυτού την εφαρμογή στο κινητό μας τηλέφωνο.

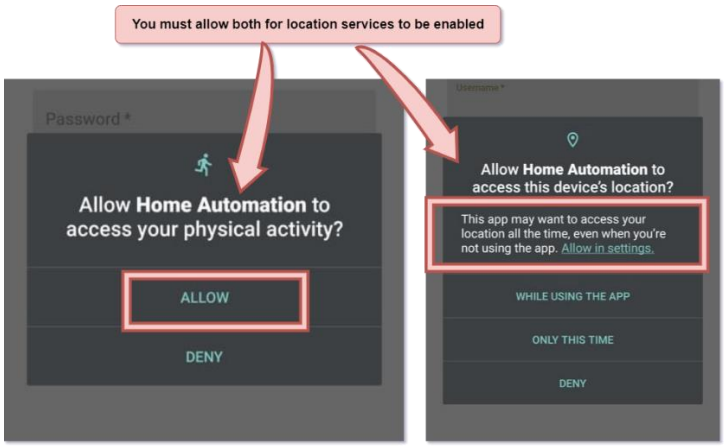
9.3 Το γραφικό περιβάλλον και ο ορισμός αυτοματισμών βάσει την τοποθεσία

Η εφαρμογή, με εξαίρεση τη προσθήκη του νέου τρόπου ορισμού αυτοματισμών, είναι πανομοιότυπη με τον web client, ενώ έχει την ίδια ακριβώς εμφάνιση, όταν αυτός χρησιμοποιείται στους mobile web browsers. Στο Σχήμα 9.1 φαίνεται το εικονίδιο για την έναρξη της εφαρμογής μέσα από το Android λειτουργικό, καθώς επίσης και η οθόνη του Dashboard που επιβεβαιώνει τη σχετική παραδοχή για την εμφάνιση της.



Σχήμα 9.1: Απεικόνιση του εικονιδίου της εγκατεστημένης εφαρμογής (αριστερά) και της Dashboard οθόνης(δεξιά)

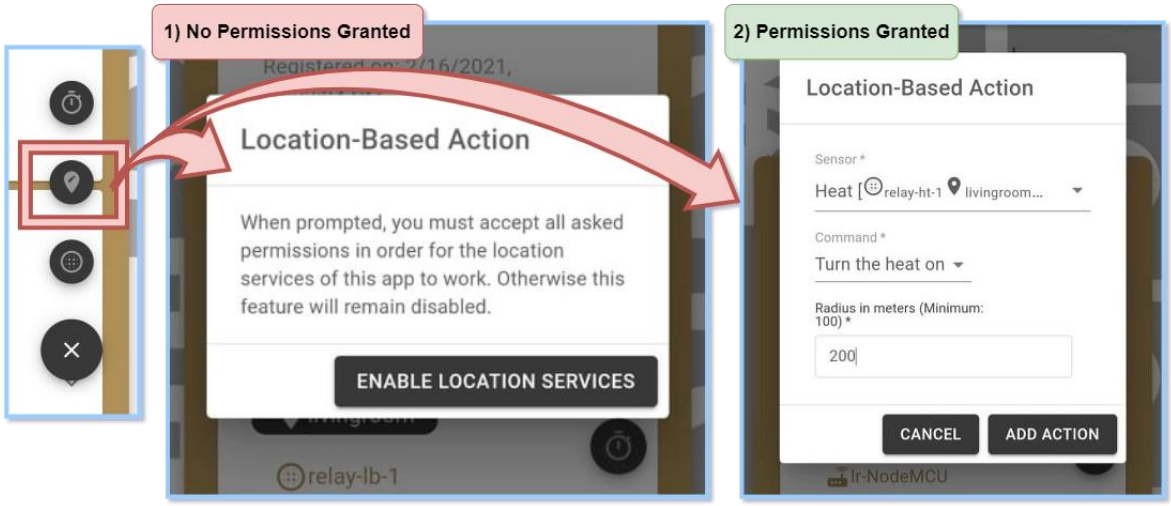
Εκκινώντας την εφαρμογή για πρώτη φορά, υποδέχονται τον χρήστη οι σχετικές ειδοποιήσεις που ζητάν την αποδοχή των σχετικών με την τοποθεσία αδειών (βλέπε Σχήμα 9.2). Ο χρήστης πρέπει να αποδεχτεί τόσο αυτή που αφορά το Physical Activity (αριστερά στο Σχήμα 9.2) όσο και αυτή για τον εντοπισμό της τοποθεσίας στο παρασκήνιο από την εφαρμογή (δεξιά στο Σχήμα 9.2). Μόνο εφόσον και οι δυο γίνουν αποδεκτές, μπορεί να οριστεί αυτοματισμός σχετικά με την τοποθεσία στη συνέχεια.



Σχήμα 9.2: Απεικόνιση της εμφάνισης των σχετικών μηνυμάτων για την παροχή αδειών από τον χρήστη προς την εφαρμογή.

Η υπηρεσία για τον ορισμό αυτοματισμού με βάση τη τοποθεσία βρίσκεται στη σελίδα των Actions και είναι προσβάσιμη μέσω του Speed Dial μαζί με τα άλλα είδη αυτοματισμών. Πατώντας την επιλογή Location-Based στο Speed Dial, ο χρήστης εμφανίζει ένα modal στην οθόνη (βλέπε Σχήμα 9.3).

1) Αν προηγουμένως δεν δεχτεί να δώσει τις σχετικές άδειες, τότε η υπηρεσία παραμένει απενεργοποιημένη και το modal ωθεί τον χρήστη να πιέσει το κουμπί ENABLE LOCATION SERVICES που θα τον οδηγήσει στις σχετικές ρυθμίσεις του λειτουργικού για την εφαρμογή, ώστε να τις ενεργοποιήσει.
2) Αν οι άδειες έχουν γίνει αποδεκτές, τότε το modal ζητά να επιλεχθεί ο αισθητήρας – actuator μαζί με την εντολή που θα εκτελέσει ο αυτοματισμός όταν ενεργοποιηθεί. Παρακάτω και σε μια σχετική φόρμα, ζητείται από τον χρήστη να πληκτρολογήσει την ακτίνα γύρω από το σπίτι του σε μέτρα. Η ακτίνα αυτή προσδιορίζει τη μέγιστη απόσταση από το σπίτι στην οποία πρέπει να βρίσκεται το κινητό του χρήστη για να ενεργοποιηθεί ο αυτοματισμός.



Σχήμα 9.3: Απεικόνιση του modal για τον ορισμό ενός Location-Based Action. 1) Όταν οι άδειες για την τοποθεσία είναι απενεργοποιημένες ή 2) όταν είναι ενεργοποιημένες.

ΚΕΦΑΛΑΙΟ 10

Η ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕΤΑΞΥ CLIENT ΚΑΙ SERVER

10.1 Δημιουργία REST API στον server

Η υλοποίηση τόσο του web client όσο και της Android εφαρμογής, εκτός από τον κώδικα για το γραφικό περιβάλλον, βασίζεται και στην επικοινωνία της με τον server. Για να είναι εφικτή η πρόσβαση της client εφαρμογής στα δεδομένα του συστήματος, αποτελεί αναγκαίο βήμα η κατασκευή ενός REST API στην πλευρά του server. Στη συνέχεια και μέσω του API αυτού, υφίστανται τα κατάλληλα endpoints, τα οποία είναι στη διάθεση του client να τα χρησιμοποιήσει για να αλληλοεπιδράσει με τον server, μέσω HTTP requests.

Ως REST API χαρακτηρίζεται η αρχιτεκτονική δομή ενός API, του οποίου κυρία λειτουργία αποτελεί η χρήση HTTP Requests για να προσπελάσει και να χρησιμοποιήσει δεδομένα. Οι μέθοδοι που ορίζουν τον τρόπο αξιοποίησης των δεδομένων αυτών μέσα από το REST API, αφορούν το διάβασμα, την προώθηση, την αλλαγή ή την διαγραφή τους και αντιστοιχούν στις πράξεις GET, POST, PUT, DELETE [30].

Στον Πίνακα 10.1 παρουσιάζεται το σύνολο όλων των endpoints που εκθέτει ο server μαζί με την ενέργεια που πραγματοποιούν, τη μέθοδο και το μονοπάτι τους. Αναλυτικά, μέσω των endpoints αυτών, η client εφαρμογή μπορεί συνολικά να ζητήσει τους ενεργούς αισθητήρες (3 στον Πίνακα 10.1), τα δεδομένα ενός εξ αυτών (4), τους αποθηκευμένους αυτοματισμούς (5) και επιπλέον να ορίσει τρία είδη Actions(7, 8, 9), να διαγράψει ένα από τα ήδη αποθηκευμένα (6) και να επικυρώσει την εγκυρότητα των στοιχείων του χρήστη(1, 2).

Πίνακας 10.1: Απεικόνιση των endpoints του REST API της εφαρμογής

No	TASK	METHOD	PATH
1	Sign in with username and password	POST	/users/signin
2	Verify login token	GET	/verifyToken
3	Get all active sensors	GET	/activeSensors
4	Get all measurements from a sensor	GET	/measurements/:sensorName
5	Get all actions	GET	/actions
6	Delete an action	DELETE	/actions/:actionid
7	Store a timer action	POST	/timerActions
8	Store a sensor-based action	POST	/sensorBasedActions
9	Store a location-based action	POST	/locationBasedActions
10	Post location coordinates	POST	/location

Για τα endpoints με αριθμό 1,2 στον Πίνακα 10.1 που αφορούν την εγκυρότητα των στοιχειών του χρήστη, η εφαρμογή του server κάνει συγκεκριμένα τα εξής:

- Ο client στέλνει τα στοιχεία που πληκτρολογήθηκαν ως HTTP Request (No. 1) στον server, την στιγμή που πιέζει το κουμπί Sign In ο χρήστης. Ο server στη συνέχεια συγκρίνει τα στοιχεία αυτά με τα σωστά και στέλνει θετική (μαζί με ένα μοναδικό token string) ή αρνητική απάντηση στον client.
- Όταν ο χρήστης πραγματοποιεί ανανέωση της σελίδας στον browser, στέλνεται ένα HTTP Request (No. 2) στον server, όπου και ελέγχεται το token που αναφέρθηκε στο πρώτο bullet. Μέσω του token αυτού, ο server καταλαβαίνει πως πρόκειται για την ίδια σελίδα του web client στον browser, οπότε και ο χρήστης εισέρχεται κατευθείαν στο Dashboard της εφαρμογής, χωρίς να πρέπει να ξανά-εισάγει τα στοιχεία του στην Login οθόνη.

10.2 Προσθήκη δυνατότητας επικοινωνίας μέσω MQTT στον client

Στο Κεφάλαιο 5 και 6 παρουσιάστηκε το MQTT πρωτόκολλο και πώς αυτό χρησιμοποιήθηκε για την ανταλλαγή μηνυμάτων μεταξύ server και sensor nodes. Στην ανάπτυξη της client εφαρμογής που παρουσιάζεται σε αυτό το Κεφάλαιο, το MQTT μπορεί να χρησιμοποιηθεί με τον ίδιο τρόπο, για τη προβολή των δεδομένων σε πραγματικό

χρόνο και χωρίς να απαιτείται η χειροκίνητη ανανέωση της σελίδας από τον χρήστη της εφαρμογής, καθώς επίσης και για την εμφάνιση ειδοποιήσεων για σχετικά με το σύστημα γεγονότα(σύνδεση/αποσύνδεση sensor node, ενεργοποίηση κάποιου action).

Παρόλα αυτά, η αξιοποίηση του MQTT σε μια ιστοσελίδα και μέσα από έναν web browser είναι σύνθετη. Για την επίτευξη της επικοινωνίας ανάμεσα σε έναν web client (στον ρόλο του MQTT Client) και τον MQTT Broker, παρεμβάλλονται τα Web Sockets. Συγκεκριμένα, ως πρωτόκολλο που επιτρέπει την αμφίδρομη επικοινωνία μεταξύ του browser και ενός web server πάνω από το TCP, τα Web Sockets επιτρέπουν την ενθυλάκωση MQTT πακέτων σε Web Socket frames και τη μεταφορά τους στο δίκτυο. Επιπλέον, απαραίτητη προϋπόθεση για να μπορεί να ολοκληρωθεί η διαδικασία, αποτελεί η υποστήριξη των Web Sockets από την υλοποίηση του MQTT Broker σε τοπικό επίπεδο [31].

Για να υφίσταται λοιπόν επικοινωνία μέσω MQTT ανάμεσα στον web client που κατασκευάζεται και τον broker του συστήματος, πραγματοποιούμε μια σειρά από βήματα που ικανοποιούν τη μεσολάβηση των Web Sockets που αναφέρθηκαν προηγουμένως.

Στην Node εφαρμογή του server, επισημάνθηκε στο Κεφάλαιο 6, η χρησιμοποίηση της JavaScript βιβλιοθήκης aedes για την κατασκευή του MQTT broker. Η ίδια βιβλιοθήκη υποστηρίζει την επικοινωνία με browsers μέσω των Web Sockets, και επομένως, με μικρές τροποποιήσεις στον κώδικα, ο broker «ακούει» πλέον πέρα από τη πύλη 1883, και στην πύλη 8883 που αντιστοιχεί στην επικοινωνία του MQTT πάνω από το πρωτόκολλο των Web Sockets (βλέπε αριστερό πλαίσιο στο Σχήμα 10.1). Από την πλευρά του web client, δίνουμε στην εφαρμογή, δυνατότητες MQTT Client με τη χρήση της βιβλιοθήκης MQTT.js [32]. Έτσι, ο web client μπορεί πλέον ως κανονικός MQTT Client να συνδεθεί στον broker, να κάνει publish μηνύματα και να κάνει subscribe σε topics. Η αφαιρετική δομή του αρχείου του κώδικα που τον ορίζει ως τέτοιον φαίνεται στο δεξί πλαίσιο του Σχήματος 10.1.

Σχήμα 10.1: Απεικόνιση μέρος του κώδικα του aedes broker μετά την ενσωμάτωση των Web Sockets (αριστερό πλαίσιο) και αφαιρετική δομή του σχετικού κώδικα του web client που του παρέχει MQTT δυνατότητες επικοινωνίας ως MQTT Client (δεξί πλαίσιο)

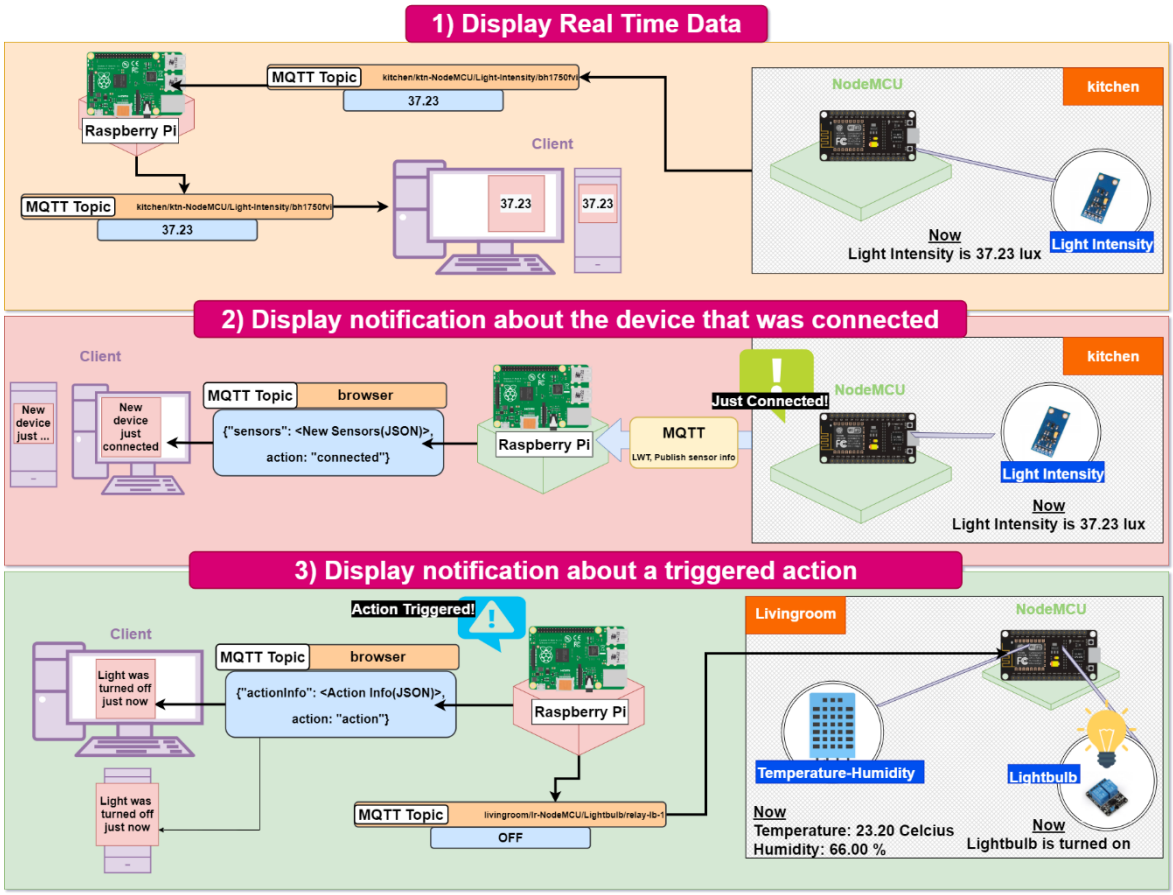
10.3 Η χρήση του MQTT στον client

Μετά και την ενσωμάτωση των MQTT δυνατοτήτων στον client, η επικοινωνία μέσω MQTT φαίνεται στο Σχήμα 10.2, όπου παρουσιάζονται τα τρία σενάρια αλληλεπίδρασης server και client μέσω του πρωτοκόλλου.

1) Στη πρώτη περίπτωση (το πάνω διάγραμμα στο Σχήμα 10.2), επισημαίνεται η προβολή δεδομένων σε πραγματικό χρόνο. Ο Web Client, κατά το άνοιγμα, στέλνει ένα HTTP request στον server για το διάβασμα όλων των αισθητήρων (No. 3 στον Πίνακα 10.1). Αφού λάβει ως απάντηση όλους τους αισθητήρες και τα στοιχεία τους, κάνει subscribe, ως MQTT Client, στα topics τα οποία κάνουν publish τα δεδομένα τους. Με τον τρόπο αυτό, μπορεί και προβάλλει την πληροφορία σε πραγματικό χρόνο.
2) Στη δεύτερη περίπτωση (το μεσαίο διάγραμμα στο Σχήμα 10.2), αναλύεται το σενάριο της σύνδεσης ενός νέου sensor node στο σύστημα. Όταν συμβαίνει αυτό, υπάρχει η επιθυμία να εμφανιστούν στην οθόνη του client, οι συνδεμένοι αισθητήρες του sensor node μαζί με τις πληροφορίες που φέρουν. Για τον σκοπό αυτό, η client εφαρμογή είναι subscriber σε ένα αποκλειστικό για αυτή topic, το

«browser». Το topic αυτό, ο broker το χρησιμοποιεί για να στείλει το είδος του γεγονότος που συνέβη, όπως η σύνδεση ενός καινούριου sensor node στο σύστημα, αποστέλλοντας ταυτόχρονα και τα στοιχεία των νέων αισθητήρων που υπάρχουν σε αυτό. Ο client, συνεπώς, παραλαμβάνει όλα τα στοιχεία που χρειάζεται για να εμφανίσει τους νέους αισθητήρες στην οθόνη και παράλληλα ειδοποιεί τον χρήστη για το γεγονός, με το σχετικό μήνυμα.

3) Η τρίτη περίπτωση (το κάτω διάγραμμα στο Σχήμα 10.2), περιγράφει την ενεργοποίηση κάποιου αυτοματισμού στο σύστημα. Όπως θα δούμε στη συνέχεια, όταν ο server του συστήματος εκτελεί έναν αυτοματισμό βασισμένο σε κάποια συνθήκη, ταυτόχρονα στέλνει στο topic «browser» το είδος της ενέργειας που πραγματοποιήθηκε, έτσι ώστε ο client να εμφανίσει μια σχετική ειδοποίηση προς τον χρήστη.



Σχήμα 10.2: Απεικόνιση των τριών περιπτώσεων χρήσης του MQTT στην client εφαρμογή του συστήματος

10.4 Η υλοποίηση των αυτοματισμών στον server

Στα Κεφάλαια 8,9 παρουσιάστηκε ο τρόπος με τον οποίον ο χρήστης ορίζει τα τρία είδη αυτοματισμών μέσα από το γραφικό περιβάλλον. Όπως αναλύθηκε, με τον επιτυχή ορισμό κάθε Action, στέλνεται ένα HTTP Request στον server για την κατοχύρωση του, ο οποίος μετά είναι υπεύθυνος για τη διαχείριση του.

Η διαχείριση των Actions γίνεται, κατά αρχήν, με την αποθήκευση τους στη βάση δεδομένων. Αυτό συμβαίνει διότι είναι επιθυμητό σε μια ενδεχόμενη επανεκκίνηση του server να υπάρχουν κατοχυρωμένοι όλοι οι αυτοματισμοί, χωρίς να πρέπει ο χρήστης να τους ξανά ορίσει. Γενικότερα, η ολική διαχείριση τους από τον server βασίζεται σε ένα module αρχείο actions.js που κατασκευάζεται στην Node εφαρμογή και υπηρετεί το σύνολο των endpoints που σχετίζεται με αυτά (βλέπε Πίνακα 10.1, No. 5,6,7,8,9,10).

Για την δρομολόγηση των αυτοματισμών που είναι είτε Timer είτε Sensor-Based Actions, το module χρησιμοποιεί μια πρόσθετη βιβλιοθήκη JavaScript, την node-schedule [33]. Μέσω της βιβλιοθήκης αυτής, μπορούμε να δρομολογήσουμε την εκτέλεση του κώδικα που επιθυμούμε σε μια συγκεκριμένη ημερομηνία και ώρα και προαιρετικά να ορίσουμε την επανάληψης της διαδικασίας ανά τα χρονικά διαστήματα που επιθυμούμε.

Το actions.js κάνει εκτενή χρήση αυτής της βιβλιοθήκης και οι βασικές λειτουργίες του αναλυτικά είναι οι εξής:

Για τα Timer Actions:

Όταν ο χρήστης προσθέτει ένα Timer Action μέσα από τον Client, ο server λαμβάνει το ανάλογο HTTP Request (βλέπε No. 6 στον Πίνακα 10.1) για τη προσθήκη του στο σύστημα. Το Action αυτό καταγράφεται στη βάση δεδομένων. Επιπλέον, για την δρομολόγηση του αυτοματισμού, το module χρησιμοποιεί τη node-schedule βιβλιοθήκη που αναφέραμε παραπάνω και δρομολογεί την εκτέλεση του αυτοματισμού με βάση τα χρονικά πλαίσια που ζήτησε ο χρήστης. Όταν φτάσει το χρονικό σημείο που ορίστηκε, το module εκτελεί τον αυτοματισμό και ειδοποιεί τον client (μέσω του MQTT όπως επισημάνθηκε παραπάνω) για να εμφανίσει την ειδοποίηση της εκτέλεσης.

Για τα Sensor-Based Actions:

Όταν ο χρήστης προσθέτει ένα Sensor-Based Action μέσα από τον client, ο server λαμβάνει το HTTP Request (βλέπε No. 7 στον Πίνακα 10.1) και το module στη συνέχεια αποθηκεύει το Action στη βάση. Για τα Sensor-Based Actions, το module χρησιμοποίει τη node-schedule βιβλιοθήκη για να ορίσει την εκτέλεση ενός ελέγχου κάθε 5 δευτερόλεπτα. Κατά την επίτευξη του ελέγχου αυτού, ελέγχεται κάθε φορά ο επιλεγμένος αισθητήρας για το αν έχει τη μέτρηση που όρισε ο χρήστης. Εφόσον οι συνθήκες ικανοποιούνται, ο αυτοματισμός εκτελείται και ενημερώνεται ο client για να εμφανίσει την ειδοποίηση της εκτέλεσης.

Για τα Location-Based Actions:

Όταν ο χρήστης προσθέτει ένα Location-Based Action μέσα από την Android εφαρμογή, και εδώ ένα HTTP Request (βλέπε No. 9 στον Πίνακα 10.1) φτάνει στον server και το module αποθηκεύει το Action στη βάση. Στο Κεφάλαιο 9 αναφέρθηκε ότι η βιβλιοθήκη cordova-plugin-background-geolocation που χρησιμοποιείται για τον εντοπισμό της τοποθεσίας, δίνει τη δυνατότητα να οριστεί ένα URL που θα στέλνονται οι μετρήσεις. Αυτό το URL, το ορίζουμε ως το No.10 endpoint του Πίνακα 10.1. Συνεπώς, κάθε φορά που οι μετρήσεις φτάνουν μέσω HTTP στο συγκεκριμένο endpoint του server, το module ελέγχει αν τα αποθηκευμένα Location-Based Actions πληρούν τις προϋποθέσεις για την εκτέλεση των αυτοματισμών του. Επιπρόσθετα, χρησιμοποιούμε τη συνάρτηση isPointWithinRadius() που είναι μέρος μιας πρόσθετης βιβλιοθήκης της JavaScript με όνομα Geolib [34] και η οποία ελέγχει αν μια τοποθεσία βρίσκεται εντός της ορισμένης ακτίνας μιας άλλης τοποθεσίας. Έχοντας ορίσει στον κώδικα την στατική τοποθεσία (γεωγραφικό πλάτος και μήκος) του σπιτιού μέσα στο οποίο είναι εγκατεστημένο το παρόν σύστημα οικιακών αυτοματισμών, το module συγκρίνει τις μετρήσεις που έρχονται για κάθε Location-Based Action με τη τοποθεσία της οικίας και στη συνέχεια πραγματοποιεί τα εξής:

- Αν ο χρήστης βρίσκεται εντός της ζητούμενης ακτίνας όταν όρισε το Action, τότε το Action ορίζεται ως «εκτελεσμένο» εξ αρχής, χωρίς όμως να εκτελείται ο αυτοματισμός.
- Όταν ο χρήστης απομακρυνθεί από την ακτίνα που έχει ορίσει για ένα αποθηκευμένο Action που εκείνη τη στιγμή θεωρείται «εκτελεσμένο», τότε αλλάζει η κατάσταση του σε «μη εκτελεσμένο» .

- Όταν ο χρήστης βρεθεί εντός της ακτίνας που έχει ορίσει σε κάποιο Action που εκείνη τη στιγμή έχει την τιμή «μη εκτελεσμένο», τότε αυτό ξανά-ορίζεται σε εκτελεσμένο και ο αυτοματισμός εκτελείται στέλνοντας τη σχετική ειδοποίηση στον client.

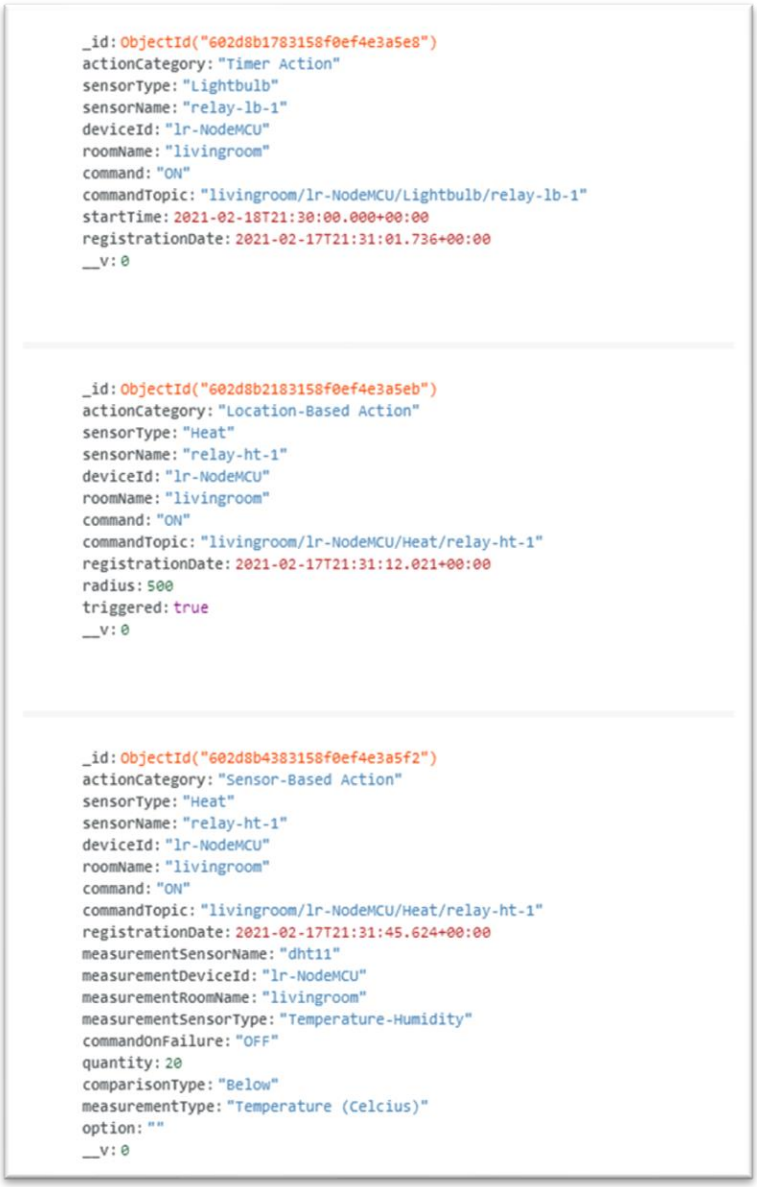
Για την ανάκτηση όλων των Actions:

Όταν ο χρήστης ανοίγει τη σελίδα των Actions στον client (βλέπε Σχήμα 8.1), ο τελευταίος στέλνει ένα HTTP Request στον server (βλέπε No. 5 στον Πίνακα 10.1) για να λάβει μια λίστα με όλα τα αποθηκευμένα Actions. Ο server μέσω του API ανακτά τα Actions από τη βάση και τα στέλνει πίσω στον client, με σκοπό αυτός να τα εμφανίσει στην οθόνη του χρήστη.

Για τη διαγραφή κάποιου Action:

Όταν ο χρήστης διαγράφει ένα Action από τη σελίδα του client, ο τελευταίος στέλνει το HTTP Request που αντιστοιχεί στη διαγραφή του Action στον server (βλέπε No. 6 στον Πίνακα 10.1) και το API στη συνέχεια σταματά την δρομολόγηση του αυτοματισμού και τον διαγράφει από τη βάση δεδομένων. Την ίδια στιγμή ο client ξαναζητά τα Actions από τον server, έτσι ώστε να ανανεώσει τη σελίδα μετά και την αλλαγή στα δεδομένα.

Στο Σχήμα 10.2 φαίνονται και τα τρία είδη Actions, έτσι όπως αποθηκεύονται στη βάση δεδομένων, μέσα από το MongoDB Atlas.



Σχήμα 10.2: Απεικόνιση μέσα από το MongoDB Atlas των τριών ειδών αυτοματισμών, έτσι όπως αυτά αποθηκεύονται στη βάση δεδομένων του συστήματος

ΚΕΦΑΛΑΙΟ 11

ΟΛΟΚΛΗΡΩΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΚΑΙ ΕΚΘΕΣΗ ΤΟΥ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

11.1 Εισαγωγή

Με το τέλος του Κεφαλαίου 10, ολοκληρώνεται η περιγραφή του συστήματος σε ότι αφορά τον κώδικα που το αποτελεί. Το σύνολο όλων των μερών από τα οποία απαρτίζεται και συγκεκριμένα ο κώδικας για τα microcontroller boards, τον κεντρικό server, τον web client και την android εφαρμογή, υπάρχει στο GitHub Repository που δημιουργήσαμε για τις ανάγκες της διπλωματική εργασίας https://github.com/paanastasiadis/home-automation-project.

Πριν το σύστημα τεθεί σε λειτουργία, το μοναδικό βήμα που απομένει για τη γενική ολοκλήρωση του, είναι η δημόσια έκθεση του. Αυτό σημαίνει πρακτικά, την δυνατότητα του χρήστη να έχει πρόσβαση στο σύστημα (μέσα από τον web client ή την android εφαρμογή) από οπουδήποτε στον κόσμο και όχι μόνο από το οικιακό του δίκτυο, αρκεί οι συσκευές του να έχουν σύνδεση στο διαδίκτυο όταν προσπαθεί να το προσπελάσει.

Εξίσου σημαντική διαδικασία για το τέλος και με σκοπό την ομαλή και αυτόματη λειτουργία του συστήματος, είναι η μετατροπή του server σε εφαρμογή που θα εκτελείται επί άπειρον ως διαδικασία παρασκηνίου στο Raspberry PI και μάλιστα αυτόματα μετά από κάθε εκκίνηση ή επανεκκίνηση της συσκευής. Αυτό πρακτικά σημαίνει ότι για να ξεκινήσει το σύστημα δεν θα πρέπει κάποιος χειριστής να τρέξει την εντολή εκκίνησης της εφαρμογής του server (συγκεκριμένα node server.js) σε κάποιον τερματικό του Raspberry Pi, αλλά αυτό θα γίνεται αυτόματα.

11.2 Ανάθεση στατικής IP και έκθεση στο διαδίκτυο

Βασικός πυλώνας της διαδικασίας είναι η ανάθεση στατικής IP στο Raspberry PI για να αποτρέψουμε μια ενδεχομένη αλλαγή της μέσα στο οικιακό δίκτυο, κάτι που θα σήμαινε τη δυσλειτουργία του συστήματος. Ακολουθώντας τον σχετικό οδηγό [35] που προσφέρεται από την επίσημη ιστοσελίδα του Raspberry Pi, ορίζουμε την IP στην μόνιμη διεύθυνση 192.168.1.69.

Στη συνέχεια, για την έκθεση της IP και άρα του server του συστήματος στο διαδίκτυο, υπάρχει η παραδοσιακή μέθοδος του Port Forwarding, η οποία θέτει δημόσια στο Διαδίκτυο μια ή περισσότερες πύλες του οικιακού δικτύου. Παρόλα αυτά, μεγάλο μειονέκτημα της ενέργειας αυτής είναι ότι θέτει το τελευταίο σε πιθανούς κινδύνους παραβίασης του. Επομένως, η έκθεση του server στον έξω κόσμο επιλέγεται να γίνει με τη χρήση της πλατφόρμας Dataplicity, η οποία μπορεί να εκθέσει έναν server δημόσια, προσπερνώντας την προηγούμενη μέθοδο. Το εργαλείο αυτό, όταν εγκατασταθεί στο Raspberry Pi, κρατά ανοιχτή μια ασφαλής HTTPS σύνδεση με τους servers της πλατφόρμας και ταυτόχρονα προσφέρει ένα URL που το αντιστοιχίζουμε στον server του συστήματος που υλοποιούμε. Στη συνέχεια, χρησιμοποιώντας το URL από οπουδήποτε στον κόσμο, αποκτούμε ασφαλή πρόσβαση στον server, αφού ένα request προς αυτόν, προωθείται πρώτα στους servers του Dataplicity, έπειτα, μέσα από την ασφαλή σύνδεση του εγκατεστημένου εργαλείου με τη πλατφόρμα και τέλος φτάνει στον προορισμό του, δηλαδή τον server του συστήματος [36, 37].

11.3 Ορισμός του server ως εφαρμογή παρασκηνίου

Τελευταίο βήμα προς την ολοκλήρωση του συστήματος είναι η μετατροπή του server σε εφαρμογή παρασκηνίου. Για να δώσουμε αυτή την λειτουργικότητα στην εφαρμογή, χρησιμοποιείται και εδώ ένα software εργαλείο, το pm2. Το pm2 είναι ένας daemon process manager για Node εφαρμογές, ο οποίος μπορεί να προσφέρει τη ζητούμενη δυνατότητα. Επιπλέον, είναι εύκολος στη χρήση του (χειρίζεται μέσω ενός απλού Command-Line Interface) και προσφέρει τη δυνατότητα εκκίνησης μιας Node εφαρμογής μαζί με την εκκίνηση του λειτουργικού, ενώ παράλληλα επιβάλλει αυτόματα την επανεκτέλεση της, σε περίπτωση που συμβεί κάποιο εσωτερικό σφάλμα σε αυτή [38].

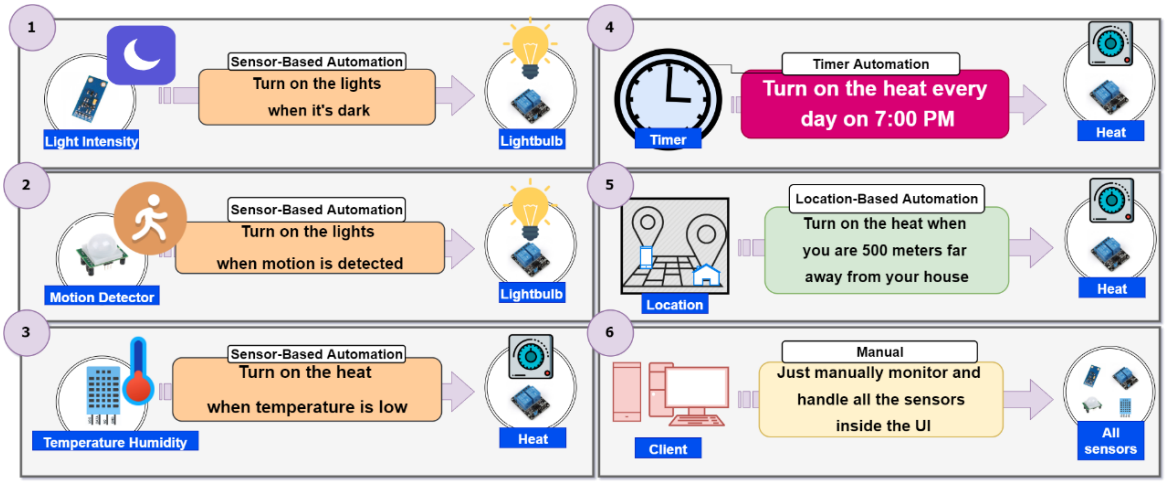
Με βάση τον οδηγό [38], χρησιμοποιούμε το CLI του pm2 για τη μετατροπή του server σε εφαρμογή παρασκηνίου, και πλέον με την έναρξη του Raspberry Pi, ξεκινά και η ομαλή λειτουργία του συστήματος οικιακών αυτοματισμών που υλοποιήσαμε.

ΚΕΦΑΛΑΙΟ 12

ΣΕΝΑΡΙΟ ΛΕΙΤΟΥΡΓΙΑΣ ΜΕ ΠΡΑΚΤΙΚΑ ΠΑΡΑΔΕΙΓΜΑΤΑ

Στο σημείο αυτό έχει επέλθει πλέον η ολοκλήρωση της ανάπτυξης και το σύστημα οικιακών αυτοματισμών τίθεται σε λειτουργία. Το είδος της αξιοποίησης που μπορεί να γίνει σε αυτό, εξαρτάται από τους διαθέσιμους αισθητήρες, αλλά και από τον τρόπο που θέλει ο χρήστης να τους χρησιμοποιεί ανάλογα με τις ανάγκες του.

Στο Σχήμα 12.1 φαίνονται μερικά πρακτικά σενάρια λειτουργίας του. Στο σενάριο 1 ορίζουμε έναν αυτοματισμό κατά τον οποίο όταν η μέρα φτάνει στις βραδινές ώρες και το φυσικό φως του ηλίου μειώνεται, ανοίγουν τα φώτα. Στο σενάριο 2, φτιάχνουμε μια μορφή προβολέα όπου ορίζουμε ότι όταν ο σχετικός αισθητήρας εντοπίσει κίνηση, τότε ανοίγει αυτόματα ο λαμπτήρας. Στα σενάρια 3,4,5, ανοίγουμε τη θέρμανση μέσω του θερμοστάτη με τρεις διαφορετικούς τρόπους: Όταν η θερμοκρασία πέσει κάτω από μία συγκεκριμένη τιμή(σενάριο 3), σταθερά και κάθε μέρα στις 7 μ.μ. (σενάριο 4) και όταν με το κινητό τηλέφωνο βρισκόμαστε σε απόσταση 500 μέτρων από το σπίτι για να προλάβει να ζεσταθεί την ώρα που θα φτάσουμε σε αυτό (σενάριο 5). Τέλος, στο σενάριο 6 αποτυπώνεται η κλασσική λειτουργία του συστήματος, όπου ο χρήστης μέσω του web client ή της Android εφαρμογής, παρακολουθεί ή δίνει εντολές στους αισθητήρες μέσα από το γραφικό περιβάλλον.



Σχήμα 12.1: Απεικόνιση μερικών από των πρακτικών σεναρίων λειτουργίας του συστήματος οικιακών αυτοματισμών

ΚΕΦΑΛΑΙΟ 13

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μετά και την ολοκλήρωση της ανάπτυξης του, τέθηκε σε λειτουργία το σύστημα οικιακών αυτοματισμών, προσφέροντας στον χρήστη μια πληθώρα ανέσεων και εκσυγχρονίζοντας την αλληλεπίδραση του με το σπίτι. Αποφεύγοντας τις ανάλογες έτοιμες λύσεις που προσφέρονται στην αγορά, η κατασκευή ενός τέτοιου συστήματος από το μηδέν, αποδεικνύει την δυνατότητα που προσφέρεται σε αυτόν που είτε διαθέτει τις κατάλληλες γνώσεις προγραμματισμού είτε έχει το πάθος να τις αποκτήσει μέσα από τη διαδικασία, να χρησιμοποιήσει τα διαθέσιμα εργαλεία που υπάρχουν για να υλοποιήσει μια ολοκληρωμένη λύση οικιακών αυτοματισμών.

Η τελική μορφή του συστήματος αποδεικνύει έμπρακτα πως κρατώντας το κόστος στα χαμηλότερα επίπεδα όσον αφορά την αγορά του απαραίτητου hardware και χρησιμοποιώντας μόνο open-source εργαλεία ή τα δωρεάν σκέλη υπηρεσιών που υπάρχουν στην αγορά, το σύστημα οικιακών αυτοματισμών καλύπτει όλους τους στόχους που είχαν τεθεί εξ αρχής. Συγκεκριμένα, είναι ένα σύστημα που η ασύρματη φύση του από την μία πλευρά δεν επιβαρύνει τον χρήστη με ενοχλητικές καλωδιώσεις, ενώ από την άλλη, δίνει μια σχετική ευελιξία ως προς την βέλτιστη οργάνωση του μέσα στην οικία. Επιπρόσθετα, η λειτουργικότητα του δεν περιορίζεται μόνο γύρω από τους αισθητήρες που παρουσιάστηκαν, αφού εφόσον το επιτάσσουν οι ανάγκες περισσότεροι από αυτούς μπορούν να αγοραστούν, και με μικρές προσθήκες στον κώδικα του συστήματος, να αναγνωρίζονται πλήρως από την εφαρμογή.

Στα αρνητικά και τις δυσκολίες που επιφέρει η κατασκευή ενός τέτοιου συστήματος είναι η αναγκαία χρήση κώδικα προγραμματισμού. Αν και το σύστημα, ως έτοιμη και εγκατεστημένη λύση, μπορεί να χρησιμοποιηθεί από τον οποιοδήποτε, η εγκατάσταση, επέκταση ή επαναρρύθμιση του, απαιτεί την παρέμβαση στον κώδικα της εφαρμογής. Σε μελλοντικές προσθήκες που θα μπορούσαν να γίνουν, η κατασκευή εύκολων στη χρήση εργαλείων που θα αναλαμβάνουν την αυτοματοποίηση των παραπάνω θα έκαναν όλες τις πτυχές του συστήματος προσβάσιμες στον οποιοδήποτε, χωρίς να απαιτείται η ανάγκη για προγραμματιστικές γνώσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] M. Asadullah and A. Raza, “An overview of home automation systems,” in *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, 2016.

[3] M. A. Matin and M. M. Islam, “Overview of wireless sensor network,” in *Wireless Sensor Networks - Technology and Protocols*, InTech, 2012.

[4] The Raspberry Pi Foundation, “Raspberry pi 4 model B specifications – raspberry pi,” *Raspberrypi.org*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Accessed: 19-Feb-2021].

[5] “Installation,” *Raspberrypi.org*. [Online]. Available: https://www.raspberrypi.org/documentation/installation/. [Accessed: 19-Feb-2021].

[6] “SSH (Secure Shell),” *Raspberrypi.org*. [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/ssh/README.md. [Accessed: 19-Feb-2021].

[7] B. Lutkevich, “What is a Microcontroller and How Does it Work?” *Techtarget.com*, 07-Nov-2019. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/microcontroller. [Accessed: 19-Feb-2021].

[8] “NodeMCU ESP8266,” *Components101.com*. [Online]. Available: https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet. [Accessed: 19-Feb-2021].

[9] “Arduino MKR1000 Wi-Fi Board,” *Components101.com*. [Online]. Available: https://components101.com/microcontrollers/arduino-mkr1000-wi-fi-board. [Accessed: 19-Feb-2021].

[10] “DHT11–temperature and humidity sensor,” *Components101.com*. [Online]. Available: https://components101.com/dht11-temperature-sensor. [Accessed: 19-Feb-2021].

[11] “HC-SR501 PIR Sensor,” *Components101.com*. [Online]. Available: https://components101.com/sensors/hc-sr501-pir-sensor. [Accessed: 19-Feb-2021].

[12] “BH1750 – Ambient Light Sensor,” *Components101.com*. [Online]. Available: https://components101.com/sensors/bh1750-ambient-light-sensor. [Accessed: 19-Feb-2021].

[13] “5V Dual-Channel Relay Module,” *Components101.com*. [Online]. Available: https://components101.com/switches/5v-dual-channel-relay-module-pinout-features-applications-working-datasheet. [Accessed: 19-Feb-2021].

[14] Adafruit Industries (2020) DHT sensor library [Source Code]. https://github.com/adafruit/DHT-sensor-library.

[15] claws (2021) BH1750 [Source code]. https://github.com/claws/BH1750http://dx.doi.org/https://github.com/claws/BH1750.

[16] “MERN Stack,” *Mongodb.com*. [Online]. Available: https://www.mongodb.com/mern-stack. [Accessed: 19-Feb-2021].

[17] “How to create a database for free,” *Mongodb.com*. [Online]. Available: https://www.mongodb.com/database/free. [Accessed: 19-Feb-2021].

[18] “MQTT - The Standard for IoT Messaging,” *Mqtt.org*. [Online]. Available: https://mqtt.org. [Accessed: 19-Feb-2021].

[19] S. Cope, “Beginner’s guide to the MQTT protocol,” *Steves-internet-guide.com*, 09-Aug-2016. [Online]. Available: http://www.steves-internet-guide.com/mqtt/. [Accessed: 19-Feb-2021].

[20] The HiveMQ Team, “MQTT topics & Best Practices - MQTT essentials: Part 5,” *Hivemq.com*. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/. [Accessed: 19-Feb-2021].

[21] moscajs (2021) Aedes [Source Code]. https://github.com/moscajs/aedes.

[22] N. O’Leary (2020) Arduino Client for MQTT [Source code]. https://github.com/knolleary/pubsubclient.

[23] The HiveMQ Team, “Last Will and Testament - MQTT Essentials: Part 9,” *Hivemq.com*. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/. [Accessed: 19-Feb-2021].

[24] “React,” *Reactjs.org*. [Online]. Available: https://reactjs.org. [Accessed: 19-Feb-2021].

[25] “Material-UI: A popular React UI framework,” *Material-ui.com*. [Online]. Available: https://material-ui.com. [Accessed: 19-Feb-2021].

[26] “Architectural overview of Cordova platform - Apache Cordova,” *Apache.org*. [Online]. Available: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. [Accessed: 19-Feb-2021].

[27] S. Patil, “Using React with Cordova,” *Medium*, 12-Aug-2019. [Online]. Available: https://medium.com/@pshubham/using-react-with-cordova-f235de698cc3. [Accessed: 19-Feb-2021].

[28] C. Scott (2020) @mauron85/cordova-plugin-background-geolocation [Source code]. https://github.com/mauron85/cordova-plugin-background-geolocation.

[29] “Access location in the background,” *Android.com*. [Online]. Available: https://developer.android.com/training/location/background. [Accessed: 19-Feb-2021].

[30] A. S. Gillis, “REST API (RESTful API),” *Techtarget.com*, 22-Sep-2020. [Online]. Available: https://searchapparchitecture.techtarget.com/definition/RESTful-API. [Accessed: 21-Feb-2021].

[31] The HiveMQ Team, “MQTT over WebSockets - MQTT Essentials Special,” *Hivemq.com*. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-special-mqtt-over-websockets/. [Accessed: 19-Feb-2021].

[32] MQTT.js (2020) MQTT.js [Source code]. https://github.com/mqttjs/MQTT.js.

[33] M. Patenaude (2021) Node Schedule [Source code]. https://github.com/node-schedule/node-schedule.

[34] M. Bieh (2020) Geolib [Source code]. https://github.com/manuelbieh/geolib.

[35] “TCP/IP networking,” *Raspberrypi.org*. [Online]. Available: https://www.raspberrypi.org/documentation/configuration/tcpip/. [Accessed: 19-Feb-2021].

[36] “How it works,” *Dataplicity.com*. [Online]. Available: https://docs.dataplicity.com/docs/how-it-works. [Accessed: 19-Feb-2021].

[37] “Host a website from your Pi,” *Dataplicity.com*. [Online]. Available: https://docs.dataplicity.com/docs/host-a-website-from-your-pi. [Accessed: 19-Feb-2021].

[38] “Quick Start,” *Pm2.keymetrics.io*. [Online]. Available: https://pm2.keymetrics.io/docs/usage/quick-start/. [Accessed: 19-Feb-2021].