Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Πλατφόρμα για Απομακρυσμένο Έλεγχο Οικίας μέσω Εφαρμογής Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Χρυσόστομος Σ. Χρίστου

Επιβλέπων: Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πλατφόρμα για Απομακρυσμένο Έλεγχο Οικίας μέσω Εφαρμογής Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Χρυσόστομος Σ. Χρίστου

Επιβλέπων: Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $1^{η}$ Ιουλίου 2016.

.....	.....	.....
Ευστάθιος Συκάς	Μιχαήλ Θεολόγου	Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2016.

.....
Χρυσόστομος Σ. Χρίστου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα διπλωματική εργασία, υλοποιήθηκε μια πλατφόρμα για απομακρυσμένο έλεγχο και παρακολούθηση οικίας. Αρχικά παρουσιάζονται θέματα σχετικά με το Internet of Things και αναλύεται θεωρητικά το πρωτόκολλο MQTT. Το MQTT είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων βασισμένο στο publish/subscribe μοντέλο και είναι το μέσο επικοινωνίας στην πλατφόρμα που υλοποιήθηκε. Για την υλοποίηση χρησιμοποιήθηκε ο υπολογιστής χαμηλού κόστους Raspberry Pi ως υπολογιστική μονάδα η οποία λαμβάνει, ερμηνεύει και επεξεργάζεται τα σήματα που στέλνουν διάφορες συσκευές τοποθετημένες στο σπίτι. Επιπλέον, το Raspberry Pi χρησιμοποιήθηκε και ως gateway της πλατφόρμας, για να προσφέρει την δυνατότητα σε διεπαφές να συνδέονται απομακρυσμένα στο οικιακό σύστημα. Επιπρόσθετα, υλοποιήθηκε μια εφαρμογή android, ως διεπαφή σε συστήματα αυτοματισμού οικίας τα οποία χρησιμοποιούν για επικοινωνία το πρωτόκολλο MQTT. Η εφαρμογή, δημιουργεί και διατηρεί πελάτες (clients), σε MQTT broker που επιλέγει ο χρήστης. Σε κάθε ένα από τους clients, ο χρήστης έχει τη δυνατότητα να ρυθμίσει γραφικό περιβάλλον στο οποίο μπορεί να παρακολουθεί και ελέγχει συσκευές που έχει εγκαταστήσει στο σπίτι του.

Λέξεις κλειδιά : Διαδίκτυο των πραγμάτων, Αυτοματισμός Οικίας, Έξυπνο σπίτι, Raspberry Pi, Android, MQTT

Abstract

This thesis, is about the development of a platform for home monitoring remotely. Initially, Internet of Things and issues that concern it are presented. Afterwards, the MQTT protocol is theoretically analysed. The MQTT is a messaging protocol based on the publish/subscribe model and is the communication protocol followed by the platform. For the implementation, Raspberry Pi, i.e. a low-cost computer, is used as a computing unit which receives, interprets and processes the signals sent by various devices installed in the house. Furthermore, the Raspberry Pi serves as a gateway for the platform, in order to communicate remotely with the home system. Additionally, an android application was implemented to act as an interface to the home automation systems that use the MQTT protocol. The application creates and maintains clients, for a MQTT broker chosen by the user. For each of the clients, the user is able to configure a graphical environment, through which he can monitor and control devices set up in the house.

Key words : Internet of Things, Home Automation, Smart Home, Raspberry Pi, Android, MQTT

Ευχαριστίες

Ευχαριστώ τον καθηγητή κ. Ευστάθιο Συκά που μου ανάθεσε την παρούσα διπλωματική εργασία καθώς και τον υποψήφιο διδάκτορα κ. Απόστολο Κοτοπούλη για την αμέριστη υποστήριξη και συμβουλές που μου πρόσφερε καθ' όλη τη διάρκεια αυτής της εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου για την άψογη συνεργασία που είχαμε αυτά τα χρόνια στο Εθνικό Μετσόβιο Πολυτεχνείο.

Τέλος, δεν θα μπορούσα να μην αναφερθώ στην πολύτιμη στήριξη των γονιών μου σε κάθε βήμα στη μέχρι τώρα σταδιοδρομία μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Πρόλογος

Η ραγδαία εξέλιξη της τεχνολογίας έχει συμβάλει στην εξέλιξη του αυτοματισμού. Ο αυτοματισμός έκανε αρχικά την παρουσία του στη βιομηχανία, αλλά η αναζήτηση του ανθρώπου για πιο άνετο τρόπο ζωής οδήγησε στην επιζήτηση χρήσης του αυτοματισμού σε προσωπικό επίπεδο.

Η εξέλιξη αυτή οδήγησε σε ένα νέο τομέα τεχνολογίας που ονομάζεται αυτοματισμός οικίας ή έξυπνο σπίτι. Αυτό σημαίνει κυρίως, ότι χρησιμοποιώντας την τελευταία λέξη της τεχνολογίας σε διάφορα λειτουργικά συστήματα στις ηλεκτρικές εγκαταστάσεις, η διαχείριση γίνεται όλο και πιο αυτοματοποιημένη και βασίζεται σε αρχές αυτοματισμού, τηλεχειρισμού, χρονοπρογραμματισμού, οπτικοποίησης κλπ.

Επιπλέον, η εξέλιξη στις φορητές συσκευές (smartphones, tablets κλπ) οδήγησε σε πιο ολοκληρωμένες εφαρμογές. Εφαρμογές που παρέχουν στους χρήστες δυνατότητες όπως να ενημερώνονται για την κατάσταση του κτιρίου μέσω κινητού τηλεφώνου ή διαδικτύου και μπορούν να επέμβουν προγραμματίζοντας κάποιο συμβάν (π.χ άναμμα θερμοσίφωνα).

Παρόλα αυτά, το κόστος υλοποίησης και εγκατάστασης έξυπνων σπιτιών, εξακολουθεί να είναι υψηλό. Το γεγονός αυτό, λειτουργεί ως αποτρεπτικός παράγοντας, για αρκετούς ανθρώπους, από την εξ ολοκλήρου κατασκευή έξυπνων σπιτιών και αρκούνται σε επιμέρους έξυπνες εφαρμογές.

1.2 Σκοπός

Λαμβάνοντας υπόψη τα πιο πάνω, αυτή η διπλωματική εργασία εκπονήθηκε με σκοπό να δείξει πως με λίγη εμπειρία σε ηλεκτρονικά και προγραμματισμό μπορεί κανείς να υλοποιήσει τις δικές του λύσεις για έξυπνες εφαρμογές, με λειτουργίες που ανταγωνίζονται προϊόντα που υπάρχουν στην αγορά. Για το σκοπό αυτό, παρουσιάζεται η διαδικασία υλοποίησης πλατφόρμας αυτοματισμού οικίας με υλικά χαμηλού κόστους και εργαλεία λογισμικού ανοιχτής-πηγής.

1.3 Δομή διπλωματικής εργασίας

Κεφάλαιο 2: Σε αυτό το κεφάλαιο γίνεται αναφορά στις M2M επικοινωνίες και στην εξέλιξή τους το Internet of Things (IoT). Επίσης, παρουσιάζεται μια από τις σημαντικότερες εφαρμογές του IoT, ο αυτοματισμός σπιτιού καθώς και το πώς οι φορητές συσκευές έχουν κατακλύσει τη ζωή μας. Επιπλέον, θίγεται και το πολύ σημαντικό ζήτημα που αφορά την ασφάλεια στις IoT εφαρμογές.

Κεφάλαιο 3: Αναφέρονται οι διάφορες τεχνολογίες υλικού και λογισμικού που χρησιμοποιήθηκαν στην υλοποίηση της πλατφόρμας οικιακού αυτοματισμού αυτής της εργασίας. Επεξηγούνται ορολογίες που χρησιμοποιούνται αργότερα στην περιγραφή της εφαρμογής.

Κεφάλαιο 4: Αναλύεται εις βάθος το πρωτόκολλο MQTT. Παρουσιάζονται θεωρητικές και πρακτικές λεπτομέρειες καθώς και τρόποι αύξησης της ασφάλειας στις εφαρμογές που έχουν ως κορμό των επικοινωνιών τους το MQTT.

Κεφάλαιο 5: Εδώ, γίνεται αναλυτική περιγραφή της πλατφόρμας που υλοποιήθηκε. Εξηγείται η αρχιτεκτονική της πλατφόρμα και οι τρόποι αντιμετώπισης σε κάθε ζήτημα που προέκυψε. Βαρύτητα δίνεται στην επεξήγηση των λειτουργιών που λαμβάνουν χώρα στον υπολογιστή Raspberry Pi.

Κεφάλαιο 6: Λεπτομερής παρουσίαση της εφαρμογής android, Hm-Monitoring, και πώς αυτή χρησιμοποιήθηκε ως διεπαφή για την πλατφόρμα αυτοματισμού οικίας.

Κεφάλαιο 7: Στο τελευταίο κεφάλαιο, συνοψίζονται σκέψεις από την υλοποίηση πλατφόρμας για απομακρυσμένο έλεγχο οικίας και παραθέτονται προτάσεις για περαιτέρω βελτίωση.

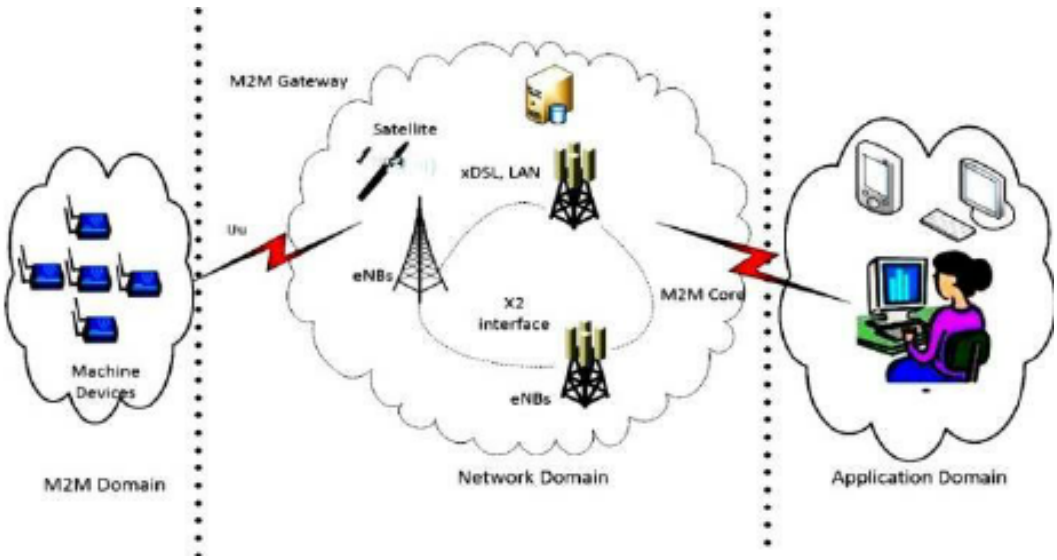
Κεφάλαιο 2

Γενικά

2.1 M2M Επικοινωνία

Ο όρος M2M (Machine to Machine, Μηχανή προς Μηχανή) επικοινωνία αναφέρεται σε αυτό που ακριβώς φαντάζεται ο αναγνώστης. Δύο μηχανές επικοινωνούν και ανταλλάζουν δεδομένα, χωρίς την ανθρώπινη παρέμβαση. Μηχανή μπορεί να είναι οποιαδήποτε ηλεκτρονική συσκευή, από ένα ολοκληρωμένο σύστημα υπολογιστή, μέχρι ένα απλό αισθητήρα. Η ραγδαία ανάπτυξη της τεχνολογίας συνετέλεσε στο να ξεφύγουμε από την ενσύρματη σύνδεση. Η ασύρματη σύνδεση δίνει τη δυνατότητα για μεγαλύτερες και πολυπλοκότερες εφαρμογές αφού είναι ευκολότερο να συνδεθούν μαζί περισσότερες μηχανές.

Ουσιαστικά τα M2M δίκτυα μοιάζουν με LAN ή WAN δίκτυα με περιορισμό στην επικοινωνία συσκευών. Αυτή η διαδικασία δίνει τη δυνατότητα στον άνθρωπο ή σε έξυπνα συστήματα να παρακολουθούν όλο το δίκτυο μέσω διεπαφής και να δίνουν εντολές σε συγκεκριμένες συσκευές. Στην εικόνα 2.1 βλέπουμε μια τυπική M2M αρχιτεκτονική.



Σχήμα 2.1: Αρχιτεκτονική M2M επικοινωνιών

Παραδοσιακά οι M2M εφαρμογές έκαναν την εμφάνισή τους στη βιομηχανία για μεταφορά πληροφοριών. Στη συνέχεια εξελίχθηκαν σε παρακολούθηση, έλεγχο και ανάλυση συσκευών ηλεκτρικού ρεύματος, φυσικού αερίου και πετρελαίου. Στη σημερινή εποχή, η χρήση τους έχει πολλαπλασιαστεί, καθιστώντας τις, την πιο ραγδαία αναπτυσσόμενη τεχνολογία για σύνδεση συσκευών στην αγορά. Ο λόγος; Εκατομμύρια συσκευές μεγάλου εύρους σε είδος μπορούν να συνδεθούν σε ένα απλό δίκτυο.

Οι μεγαλύτερες κατηγορίες στις οποίες συναντά κανείς M2M εφαρμογές στις μέρες μας είναι η βιομηχανία, η υγεία, το έξυπνο ηλεκτρικό δίκτυο και το έξυπνο σπίτι. Η χρήση στα εργοστάσια προσανατολίζεται στον έλεγχο της γραμμής παραγωγής, την αυτοματοποίηση διαδικασιών καθώς και την ασφάλεια. Στο τομέα της ιατρικής περίθαλψης συναντάμε κυρίως εφαρμογές παρακολούθησης ασθενών κατά την απουσία ιατρών π.χ παροχή οξυγόνου σε ασθενή όταν αυτό πέσει κάτω από κάποιο όριο. Επιπλέον, η διαχείριση του ηλεκτρικού δικτύου γίνεται πιο εύκολα και αποδοτικά με αυτόματη παρακολούθηση κατανάλωσης σε κάθε σημείο. Τέλος άμεση θετική επίπτωση στην ποιότητα ζωής μας έχουν οι οικιακές M2M εφαρμογές, οι οποίες μας επιτρέπουν τον απομακρυσμένο έλεγχο και την παρακολούθηση κάθε ηλεκτρικής συσκευής.

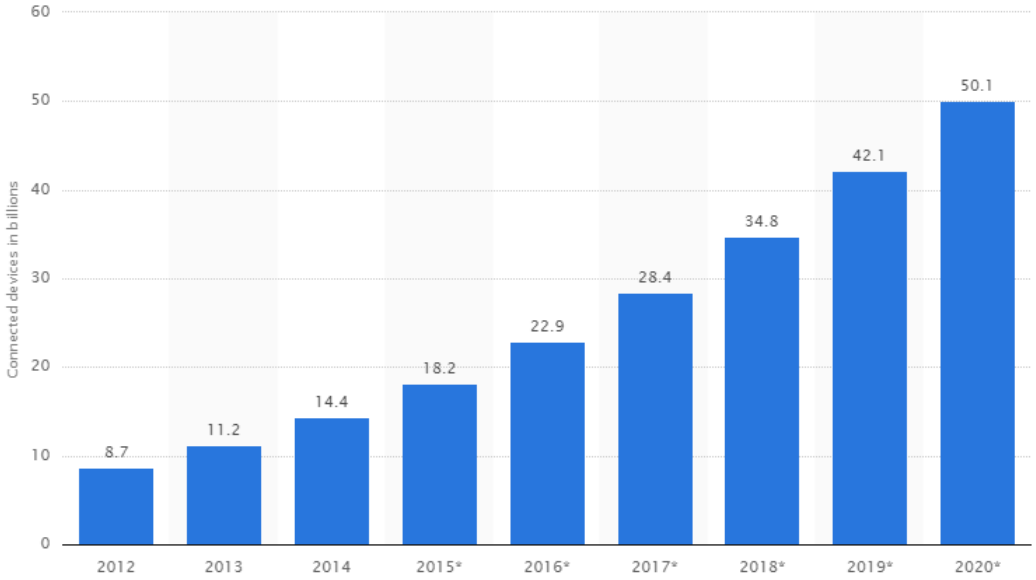
2.2 Internet of Things

Τα τελευταία χρόνια η αξία του M2M έχει αναγνωριστεί σημαντικά με αποτέλεσμα να εξελιχθεί σε αυτό που ονομάζουμε σήμερα διαδίκτυο των πραγμάτων - Internet of Thinqs (IoT). Το IoT έρχεται να καλύψει το κενό ανάμεσα σε συσκευές - αισθητήρες και δίκτυα επικοινωνιών. Μια IoT πλατφόρμα είναι ένα δίκτυο από φυσικά "πράγματα" στα οποία επιτρέπει να μαζεύουν και να ανταλλάζουν δεδομένα. Παρέχει διεπαφές για χρήστες ή αυτοματοποιημένα συστήματα για έλεγχο και παρακολούθηση και εμπλέκει πρωτόκολλα επικοινωνίας τα οποία οδηγούν στην ολοκλήρωση προχωρημένων εφαρμογών. [1]

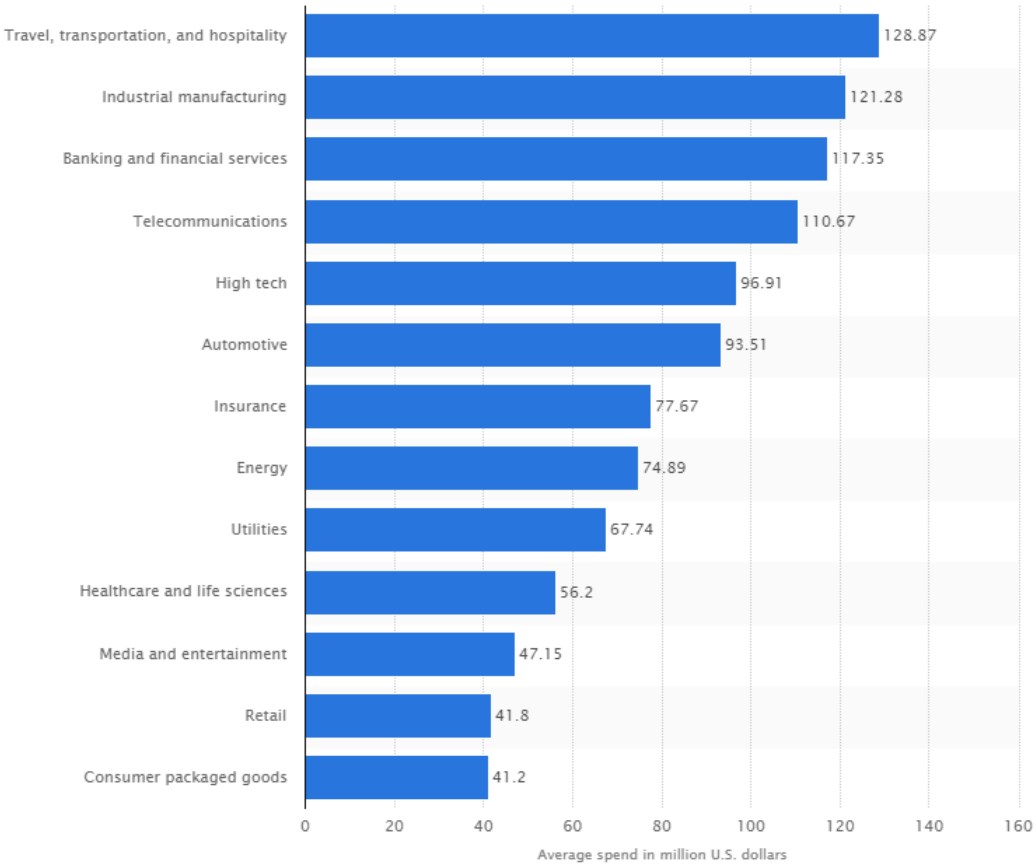
Το IoT αποτελεί σημείο αναφοράς και έρευνας, τόσο από τη βιομηχανική όσο και από την ακαδημαϊκή κοινότητα παγκοσμίως. Το αντικείμενό του απασχολεί διεθνώς πολλές ανερχόμενες εταιρείες, ακαδημαϊκά ινστιτούτα, κυβερνητικές οργανώσεις και μεγάλες επιχειρήσεις. Εύστοχα μπορεί να το χαρακτηρίσει κανείς ως το «Διαδίκτυο των Πάντων» αφού υπόσχεται διασύνδεση μεταξύ ανθρώπων και αντικειμένων οποιαδήποτε στιγμή, σε οποιοδήποτε μέρος χρησιμοποιώντας το Διαδίκτυο. Κατά συνέπεια, συμβάλλει στη δημιουργία ενός καλύτερου κόσμου για τους ανθρώπους, όπου οι έξυπνες συσκευές θα μπορούν κάθε στιγμή να λαμβάνουν γνώση τι θέλει ο άνθρωπος και πώς το θέλει.

Για να αναλογιστεί κανείς την αξία αυτής της εξέλιξης παρουσιάζεται στο σχεδιάγραμμα 2.2 η αναμενόμενη αύξηση των συνδεδεμένων συσκευών, ξεπερνώντας τις 50 δισεκατομμύρια συσκευές παγκόσμιος το 2020. Ακόμη όλο και περισσότεροι τομείς της αγοράς αναπτύσσουν τις επιχειρήσεις τους ξοδεύοντας πολλά χρήματα σε IoT εφαρμογές. Στο διάγραμμα 2.3 μπορεί να δει κανείς σε μέσο όρο πόσα ξοδεύει κάθε τομέας της αγοράς. [2]

Επιπλέον, μια έρευνα από την Ericsson Mobility Report προβλέπει ότι το IoT θα έχει ξεπεράσει την κινητή τηλεφωνία μέχρι το 2018.[5]



Σχήμα 2.2: IoT - Συνδεδεμένες συσκευές

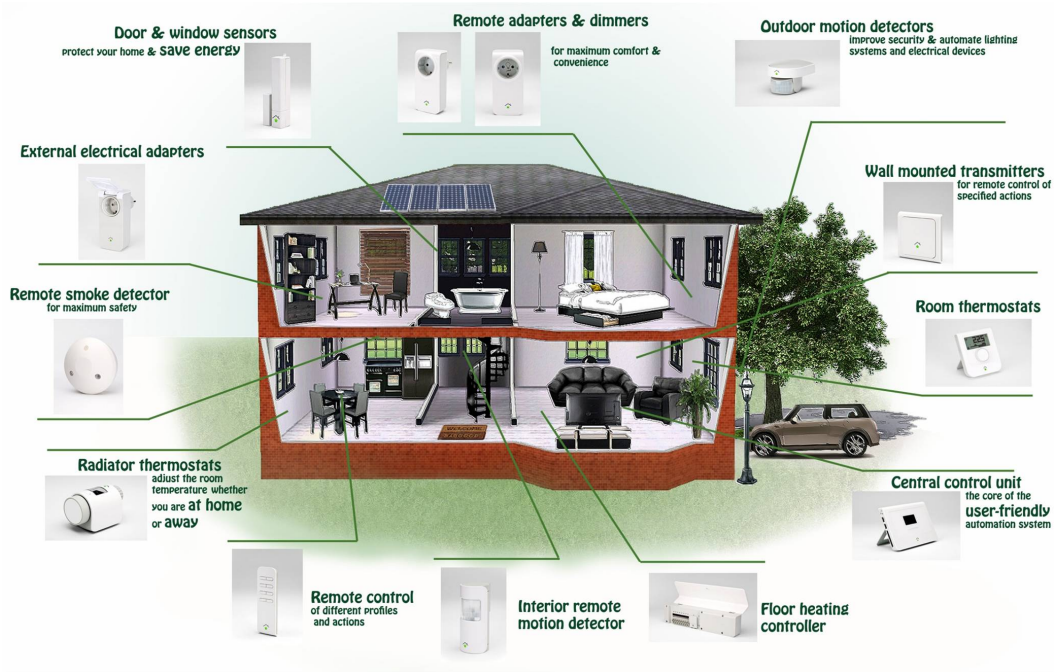


Σχήμα 2.3: IoT - Χρήματα που ξοδεύονται σε κάθε τομέα τις αγοράς

2.3 Αυτοματισμός Σπιτιού (Home Automation)

2.3.1 Αυτοματισμός Σπιτιού

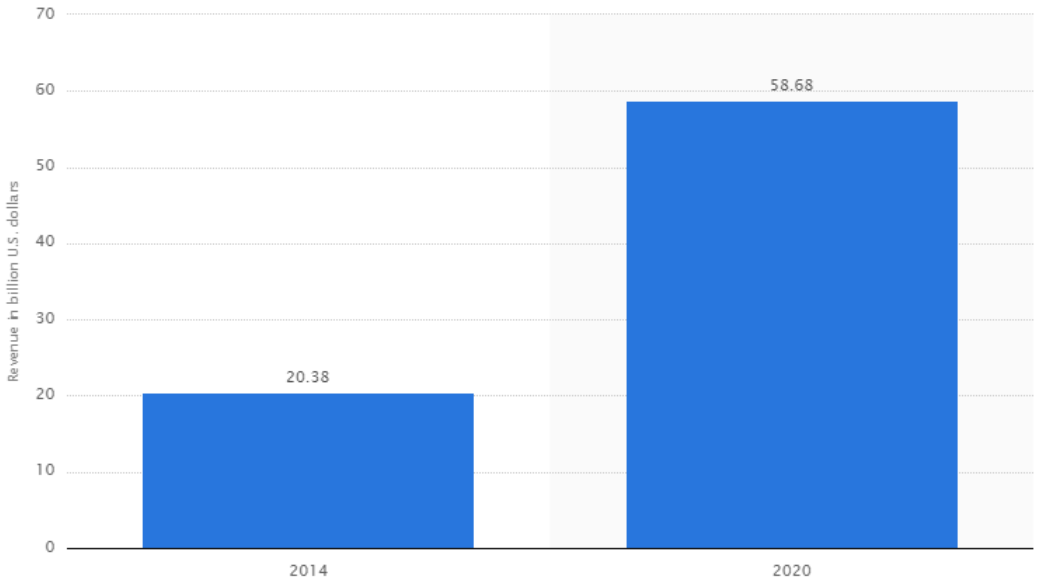
Οι έννοιες αυτοματισμός σπιτιού (Home Automation), έξυπνο σπίτι (smart home) και Domotics συχνά αναφέρονται στο ίδιο πράγμα. Αφορά την τεχνολογία που χρησιμοποιεί υπολογιστές, smartphones, και μικροεπεξεργαστές για να ελέγχει και να παρακολουθεί οικιακές συσκευές, καθώς και ότι άλλο μπορεί κανείς να φανταστεί, όπως θέρμανση, ηλεκτρική κατανάλωση, φωτισμό, παράθυρα, πόρτα του γκαράζ, κίνηση στο σπίτι και συστήματα συναγερμού. Αυτές οι τεχνολογίες μπορεί στο παρελθόν να έμοιαζαν πολυτέλεια, αλλά το σημερινό status quo τις καθιστά αναγκαίες στην καθημερινότητα μας. Ακόμη, ένα καλό σύστημα αυτοματισμού μπορεί να προσφέρει απλότητα, αύξηση βιοτικού επιπέδου, διασκέδαση, αποδοτικότητα στην κατανάλωση πόρων (ηλεκτρικό ρεύμα, πετρέλαιο θέρμανσης) και ασφάλεια για όλη την οικογένεια.



Σχήμα 2.4: Smart Home

Ιστορικά ο αυτοματισμός σπιτιού ξεκίνησε γύρω στο 1900 με την παροχή ηλεκτρικού ρεύματος στα σπίτια. Εκείνη την περίοδο άρχισαν να κάνουν την εμφάνιση τους οι βραστήρες νερού (1889), τα πλυντήρια ρούχων (1904) και πολύ σύντομα τα ψυγεία, τα στεγνωτήρια ρούχων, το σίδερο ρούχων καθώς και πλυντήρια πιάτων.[3] Στην συνέχεια, το 1975 αναπτύχθηκε η 1^{η} τεχνολογία δικτύων γενικού σκοπού για αυτοματισμό σπιτιού, X10. Το X10 είναι ένα πρωτόκολλο επικοινωνίας για ηλεκτρικές συσκευές. Η πρωταρχική του χρήση είναι η ενσύρματη μεταφορά ηλεκτρικών σημάτων σε ράδιοσυχνότητες(RF) για αναπαράσταση ψηφιακής πληροφορίας. Το X10 παραμένει μέχρι σήμερα δημοφιλής επιλογή για έξυπνες οικιακές εφαρμογές. [4]

Στις μέρες μας πολλά σπίτια κάνουν χρήση έξυπνων εφαρμογών, όπως αυτόματη ρύθμιση θέρμανσης, αυτόματο άναμμα φωτισμού στη παρουσία ανθρώπων (κίνηση) και συστήματα ασφαλείας-παρακολούθησης. Παρόλα αυτά, η έλλειψη ενός ενιαίου και απλουστευμένου πρωτοκόλλου καθώς και το υψηλό κόστος της εγκατάστασης, απωθεί το μέσο-καταναλωτή από την κατασκευή έξυπνων σπιτιών με ολοκληρωμένα συστήματα. Στα επόμενα χρόνια όμως αναμένεται σημαντική εξέλιξη σε αυτό το τομέα. Σύμφωνα με στατιστικά η αξίας της αγοράς αναμένεται σχεδόν να τριπλασιαστεί από το 2014 στο 2020, Σχ. 2.5.



Σχήμα 2.5: Smart Home - Αξία αγοράς

2.3.2 Αρχιτεκτονική Έξυπνου Σπιτιού

Με τη ραγδαία ανάπτυξη της τεχνολογίας υπάρχει δυνατότητα αυτοματισμού σχεδόν κάθε λειτουργίας συσκευών και όχι μόνο, που συμβαίνει στο σπίτι. Σε αυτές τις συσκευές έρχεται να προστεθεί η M2M επικοινωνία που αναλύσαμε πιο πάνω, καθώς οι συσκευές και συνάμα οι επικοινωνίες αυξάνονται έρχεται η αναγκαιότητα για την ύπαρξη ενός **Gateway**.

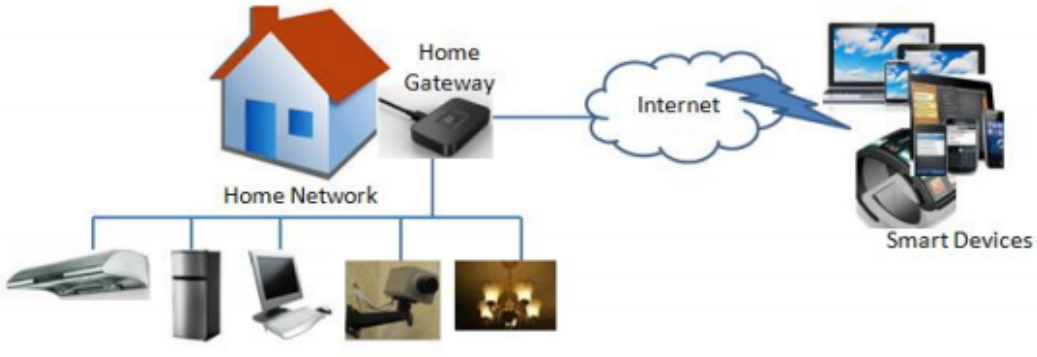
Ο ρόλος του Gateway είναι αρκετά σημαντικός, τόσο για την σύνδεση των συσκευών στο εσωτερικό δίκτυο, όσο και για την σύνδεση του εσωτερικού δικτύου με άλλα δίκτυα, κυρίως στο διαδίκτυο. Πρέπει να αλληλεπιδρά έξυπνα με εσωτερικές συσκευές και ετερογενή υποδίκτυα που πιθανόν να κάνουν χρήση διαφορετικών πρωτοκόλλων επικοινωνίας.

Τα **πρωτόκολλα επικοινωνίας** είναι από τα σοβαρότερα θέματα που έχει να αντιμετωπίσει κανείς αφού υπάρχουσες συσκευές στο σπίτι μπορεί να μην είναι συμβατές με κοινά πρωτόκολλα, καθιστώντας την επικοινωνία δύσκολη. Άρα καλό θα είναι να γίνεται καλή προεργασία και έρευνα στις έξυπνες συσκευές που θα αγοράσουμε. Πλέον μεγαλύτερη βαρύτητα δίνεται στην ασύρ-

ματη επικοινωνία για ευκολότερη εγκατάσταση συσκευών με τα επικρατέστερα πρωτόκολλα να είναι το Z-Wave και ZigBee.

Τέλος, αυτό που έρχεται να μας προσφέρει την δυνατότητα εύκολης επέμβασης για έλεγχο και παρακολούθησης είναι οι διεπαφές. Μια **διεπαφή** μπορεί να είναι ένα μια εφαρμογή σε ένα smartphone, μια ιστοσελίδα ή και πιο συγκεκριμένος πίνακας ελέγχου π.χ για ρύθμιση αυτόνομης θέρμανσης.

Ένα ολοκληρωμένο σύστημα αυτοματισμού σπιτιού όπως περιγράφεται πιο πάνω είναι ουσιαστικά μια IoT εφαρμογή.



Σχήμα 2.6: Smart Home - Συνήθης Αρχιτεκτονική

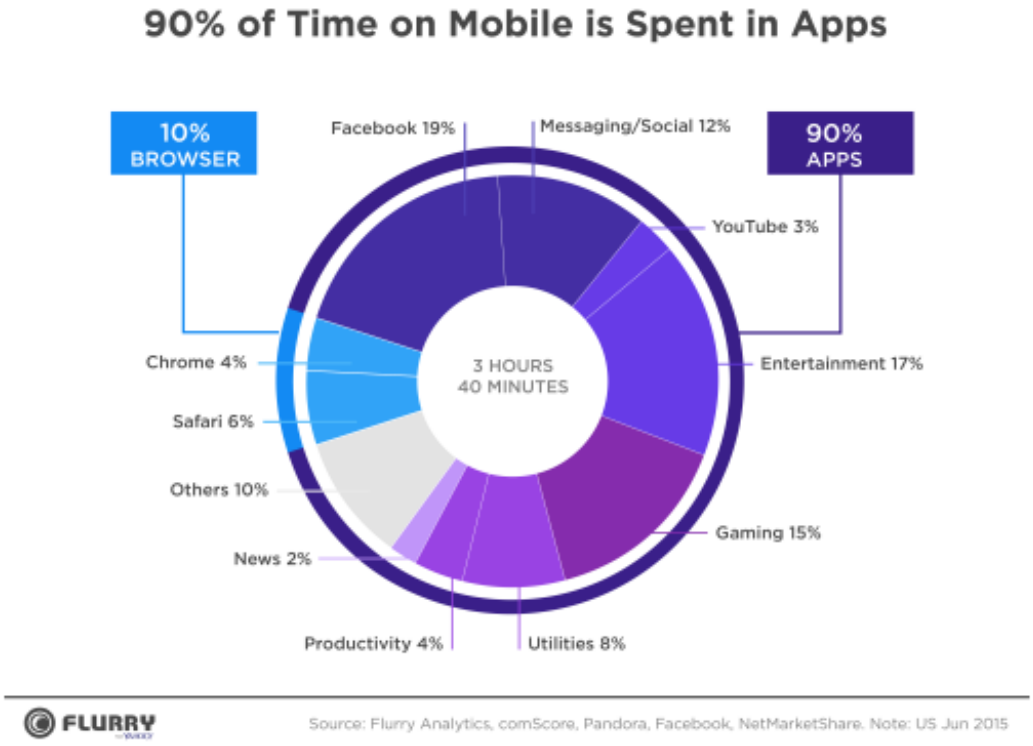
2.4 Smartphones και Tablets στη ζωή μας

Πριν από 10 χρόνια σχεδόν, δεν υπήρχαν τα smartphones και σήμερα αποτελούν αναπόσπαστο κομμάτι στην καθημερινότητα του ανθρώπου. Δεν περνάει ώρα της ημέρα που να μην χρησιμοποιούνται. Οι φορητές συσκευές και το φορητό internet έχουν αλλάξει τον τρόπο ζωής του και τον τρόπο που συνδέεται με άλλους ανθρώπους. Συγκλονιστικό για κάποιους αλλά πραγματικό είναι το γεγονός ότι υπάρχουν περισσότερα σημεία ασύρματης σύνδεσης (wireless hotspots) παρά δέντρα στην επιφάνεια της γης, καθώς και ότι πολύ σύντομα περισσότεροι άνθρωποι θα έχουν πρόσβαση στο διαδίκτυο παρά σε πόσιμο νερό.

Το smartphone έχει γίνει πλέον προέκταση του ανθρώπινου χεριού. Οι περισσότερες μελέτες δείχνουν να υπάρχει μεγάλη χρήση εν ώρα ταξιδιού, σε χώρους αναμονής, και σε διαλείμματα. Σημαντική χρήση γίνεται ακόμη και όταν βρισκόμαστε με παρέα για φαγητό, καφέ, σινεμά κλπ. Το 68% του χρόνου χρήσης αναλογεί σε χρόνο που βρίσκονται στο σπίτι. Στο χρόνο αυτό τα tablets φαίνεται να είναι προτιμότερα λόγω μεγέθους (μεγαλύτερα από smartphones αλλά μικρότερα από υπολογιστή)

Τί είναι αυτό που κρατά τόσο στενά προσκολλημένους τους χρήστες στις φορητές συσκευές; Γύρω στις 3 ώρες και 40 λεπτά κατά μέσο όρο ημερησίως αφιερώνονται στην ενασχόληση με φορητές συσκευές. Σύμφωνα με έρευνα του Flurry Analytics το 90% αυτού του χρόνου κατανέμεται στην ε-

νασχόληση με διάφορες εφαρμογές, όμως παρακολούθηση βίντεο, παιχνίδια, αγορές σε online βιτρίνες καθώς επίσης και τα μέσα κοινωνικής δικτύωσης (facebook, twitter κλπ). Το υπόλοιπο 10% αποτελεί το σερφάρισμα στο διαδίκτυο. Να σημειωθεί ότι στα πιο πάνω δεν συμπεριλαμβάνεται ο χρόνος στα συνήθη τηλεφωνήματα.[6]



Σχήμα 2.7: Κατανομή χρόνου στην ενασχόληση με φορητές συσκευές

Οι υπηρεσίες στα smartphones εξελίσσονται με γοργούς ρυθμούς. Ποιοι είναι όμως αυτοί που τα χρησιμοποιούν και παραμένουν ενημερωμένοι με όλες τις τελευταίες εξελίξεις; Αρχικά φαίνεται πως αγκαλιάζονται από διάφορες κοινωνικές ομάδες, έφηβους, νέους, γονείς αλλά και από άτομα μεγαλύτερης ηλικίας με το κάθε ένα να χρειάζεται το δικό του χρόνο προσαρμογής. Παρόλα αυτά το 50% των κατόχων φαίνεται να κάνουν χρήση μόνο των πρωταρχικών δυνατοτήτων των "έξυπνων τηλεφώνων", δηλαδή για τηλεφωνία και μηνύματα, ενώ λιγότερο από 5% επιδεικνύουν εισιτήρια μέσων μεταφοράς, συναυλιών και άλλους κωδικούς από τις φορητές τους συσκευές.

Τέλος, η Cisco κάνει ακόμη πιο δραματικές προβλέψεις, υποστηρίζοντας ότι ο μέσος άνθρωπος μέχρι το 2020 θα διατηρεί 130 terabytes προσωπικών δεδομένων σε smartphones, tablets και προσωπικούς υπολογιστές, πρόβλεψη με την οποία εξυπακούεται ότι η συμμόρφωση με νέες τεχνολογίες θα είναι αναγκαία.[7]

2.5 Ασφάλεια στις συνδέσεις

Τελευταίο, αλλά όχι έσχατο, θίγεται σε αυτό το κεφάλαιο ένα πολύ σοβαρό ζήτημα στο οποίο συνήθως δεν δίνεται η απαραίτητη προσοχή. Η ασφάλεια στις συνδέσεις και τις ανταλλαγές δεδομένων με διάφορες συσκευές. Μπορεί το IoT με πρωταρχική εφαρμογή τον οικιακό εξοπλισμό να έχει σημειώσει μεγάλη ακμή αυτοματοποιώντας πολλές καθημερινές εργασίες, αλλά υπάρχουν θέματα ασφάλειας που αν παραμελήσει ο χρήστης ίσως βρεθεί σε μη ευχάριστες καταστάσεις.

Εύκολα αντιλαμβάνεται κανείς ότι ένα έξυπνο σπίτι που βρίσκεται 24/7 συνδεδεμένο στο διαδίκτυο δίνει άπλετο χρόνο σε χάκερς να βρουν ευάλωτα σημεία στο σύστημα. Ωραία είναι η ιδέα να κοινοποιεί μια οικογένεια τι κάνει στο σπίτι και ποιες είναι οι κινήσεις της, αλλά αυτή η πληροφορία στα χέρια κάποιου ληστή δεν θα μας ήταν κάτι ευχάριστο. Απλοί χρήστες οικιακού αυτοματισμού, χωρίς τις απαραίτητες γνώσεις, είναι απίθανο να παρακολουθούν τι συμβαίνει στο δίκτυο τους και όταν αυτό είναι εκτεθειμένο, οι επιτιθέμενοι είναι δύσκολο να εντοπιστούν. Από την άλλη δεν μπορεί να αναμένεται ότι τα παιδιά ή οι φιλοξενούμενοι θα είναι προσεκτικοί με την ασφάλεια του δικτύου, εάν πρώτα δεν έχει δοθεί προσοχή κατά την εγκατάσταση. Συσκευές όπως κλειδαριές, συστήματα θέρμανσης, ψυγεία κλπ δεν είναι επικίνδυνα αλλά τα δεδομένα αλλάζουν όταν αυτά συνδεθούν στο δίκτυο. Ένα σύστημα οικιακού αυτοματισμού πρέπει να είναι όσο πιο "αόρατο" γίνεται σε άγνωστους, αλλιώς σκεφτείτε πόσους κινδύνους ελλοχεύει η γνώση της ρουτίνας του καθενός σε λάθος χέρια.

Πρέπει να προσέξει κανείς κάποια σημαντικά σημεία όταν σχεδιάζει μια ασφαλή σύνδεση. Αρχικά ένα ασφαλές και έμπιστο φυσικό δίκτυο ή VPN είναι ένας τρόπος παροχής έμπιστης σύνδεσης. Επιπλέον, σε επίπεδο εφαρμογής πρέπει να υπάρχει κάποιος έλεγχος για το ποιος θεωρείται εξουσιοδοτημένος-έμπιστος χρήστης και ποιος όχι. Καλό θα είναι να υπάρχει διαχειριστής όπου θα παρέχει σε χρήστες κωδικό και όνομα, ώστε να μην μπορεί να συνδεθεί κάποιος άγνωστος. Αμέσως μετά πρέπει να προσέξουμε την πληροφορία που ανταλλάζουν οι συσκευές μας, ιδιαίτερα μέσω διαδικτύου. Πιο εξειδικευμένοι χάκερς μπορούν να παρακολουθήσουν την επικοινωνία που έχουν οι συσκευές μας χωρίς να συνδεθούν σε αυτές και να υποκλέψουν έτσι σημαντικά δεδομένα. Για να αποτραπεί αυτό γίνεται χρήση σύγχρονων μεθόδων κρυπτογραφίας στα δεδομένα. Περαιτέρω καλές πρακτικές είναι να απομονώνουμε τη σύνδεση κάθε συσκευής όσο το περισσότερο καθώς και να αναβαθμίζουμε το σύστημά μας με τις τελευταίες τεχνολογίες.[8]

Κεφάλαιο 3

Τεχνολογίες

3.1 Raspberry Pi

Το Raspberry Pi είναι ένας υπολογιστής χαμηλού κόστους (περίπου 35 € στην Ελλάδα) σε μέγεθος πιστωτικής κάρτας. Η ιδέα ξεκίνησε το 2006 από τους Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft στο εργαστήριο υπολογιστών του πανεπιστημίου του Cambridge στην Αγγλία με αφορμή την ανησυχία για τη σταδιακή πτώση των δεξιοτήτων των μαθητών που συμμετείχαν στα μαθήματα της πληροφορικής κάθε χρόνο (Raspberry Pi Foundation, 2014). Πάνω σε αυτή την ιδέα ιδρύθηκε το Raspberry Pi Foundation,το οποίο έχει στόχο να δώσει σε όσο το δυνατόν περισσότερα παιδιά κυρίως, αλλά και ενήλικες, τη δυνατότητα να ασχοληθούν με το προγραμματισμό και να αξιοποιήσουν την επιστήμη της πληροφορικής.[11]

Σχήμα 3.1: Raspberry Pi 2

Το Pi λειτουργεί όπως ένας κλασσικός υπολογιστής˙ ο χρήστης μπορεί να πλοηγείται στο ίντερνετ, να παρακολουθήσει βίντεο υψηλής ανάλυσης, να χρησιμοποιήσει κειμενογράφο και υπολογιστικά φύλλα και φυσικά να μάθει γλώσσες προγραμματισμού, όπως η Scratch και η Python. Το πρώτο μοντέλο ήταν το A, ακολούθησαν τα B, B+ και το Φεβρουάριο του 2016 κυκλοφόρησε Raspberry Pi το 3.

Παρά τον ελάχιστον όγκο του, το Raspberry Pi στη μεγαλύτερή του έκδοση διαθέτει τετραπύρηνο επεξεργαστή 1200MHz, διπύρηνη κάρτα γραφικών, 1GB RAM, τέσσερις θύρες USB, έξοδο HDMI, τροφοδοτείται μέσω Micro USB, και

40 pins γενικής χρήσης για σύνδεση με άλλα ηλεκτρονικά και περιφερειακά. Στη εικόνα 3.2 παρουσιάζουμε για σύγκριση τα χαρακτηριστικά των τελευταίων δύο μοντέλων.

	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B
Processor Chipset	Broadcom BCM2837 64Bit Quad Core Processor powered Single Board Computer running at 1.2GHz	Broadcom BCM2836 32Bit Quad Core Processor powered Single Board Computer running at 900MHz
GPU	Videocore IV	Videocore IV
Processor Speed	QUAD Core @1.2 GHz	QUAD Core @900 MHz
RAM	1GB SDRAM @ 400 MHz	1GB SDRAM @ 400 MHz
Storage	MicroSD	MicroSD
USB 2.0	4x USB Ports	4x USB Ports
Max Power Draw/voltage	2.5A @ 5V	1.8A @ 5V
GPIO	40 pin	40 pin
Ethernet Port	Yes	Yes
WiFi	Built in	No
Bluetooth LE	Built in	No

Σχήμα 3.2: Raspberry Pi 3 vs Pi 2

Στο Raspberry Pi χρησιμοποιείται η μέθοδος SoC (System on a Chip), κατά την οποία τοποθετούνται όλα τα απαραίτητα ηλεκτρονικά, συμπιεσμένα σε ένα πολύ μικρό πακέτο, για την λειτουργία ενός υπολογιστή σε ένα μόνο chip. Αντίθετα οι κοινοί υπολογιστές έχουν ξεχωριστά chips για CPU, GPU, USB controller, RAM και άλλα.[12]

Εδώ είναι εντυπωσιακό, το πώς αυτή η υπολογιστική ισχύς συγκεντρώνεται σε τόσο λίγο χώρο και με τόσο χαμηλό κόστος. Το ερώτημα είναι το τι μπορούμε να κάνουμε με το Raspberry Pi. Όπως αποδεικνύεται, με μεράκι και φαντασία οι εφαρμογές του Raspberry Pi είναι πρακτικά απεριόριστες. Κατ' αρχάς, συνδέοντάς το σε μια οθόνη και προσθέτοντας πληκτρολόγιο και ποντίκι, έχουμε έναν πλήρη υπολογιστή, ο οποίος υποστηρίζει συγκεκριμένες διανομές

Linux. Το σύνηθες λειτουργικό σύστημα που χρησιμοποιεί είναι το Raspbian, το οποίο είναι ειδικά σχεδιασμένο για να χρησιμοποιηθεί με το Raspberry Pi. Το λειτουργικό σύστημα αποθηκεύεται στην SD Card.

Μία από τις πιο διαδεδομένες πρακτικές εφαρμογές του Raspberry Pi είναι ως ένα Media Center PC για να παίζει ταινίες στην τηλεόρασή μας, χάρη στο OSMC που είναι βασισμένο στο Kodi (πρώην XBMC). Ακόμη οι λάτρεις του Minecraft, μπορούν να χρησιμοποιήσουν το Raspberry Pi αποκλειστικά για αυτό, με την ειδική έκδοση Minecraft Pi edition. Επιπλέον όσοι ενδιαφέρονται για το retro gaming, μπορούν να φορτώσουν emulators και να έχουν ένα μηχάνημα με αμέτρητα παιχνίδια του NES, SNES, Megadrive, ή ακόμα και Arcade.

Συνδέοντας έναν εξωτερικό δίσκο πάνω, μπορεί κανείς να μετατρέψει το Raspberry Pi σε ένα αποκλειστικό σύστημα για κατέβασμα Torrent όλο το 24ωρο, χωρίς να σπαταλά πόρους του προσωπικού υπολογιστή. Το Raspberry Pi έχει κατανάλωση μόλις 4W, οπότε μπορεί να είναι σε λειτουργία νυχθημερόν. Επιπρόσθετα, με τον εξωτερικό δίσκο, μπορεί να στηθεί ένα προσωπικό Cloud service, ώστε να μπορεί να έχει πρόσβαση στα αρχεία του ο χρήστης μέσω Internet, από οπουδήποτε στον κόσμο, και να μην χρειάζεται υπηρεσίες όπως το Dropbox ή το Google Drive. Είναι επίσης απόλυτα εφικτό να στηθεί ένας οικιακός Web Server, για να ανεβάσει ο χρήστης τις δικές του ιστοσελίδες.

Τέλος με τα κατάλληλα πρόσθετα εξαρτήματα (μικροεπεξεργαστές, αισθητήρες κλπ) σε συνδυασμό με τα ήδη υπάρχοντα GPIO pins μπορούν να δημιουργηθούν ολοκληρωμένες εφαρμογές οικιακού αυτοματισμού.

3.2 Αισθητήρες και εξαρτήματα

3.2.1 Αισθητήρας Θερμοκρασίας DS18B20

Το DS18B20 είναι ένας ψηφιακός αισθητήρας θερμοκρασίας που παρέχει 9-bit έως 12-bit μετρήσεις στην κλίμακα Κελσίου και επικοινωνεί μέσω του 1-Wire διαύλου, ο οποίος εξ'ορισμού απαιτεί μόνο μια γραμμή, με ένα κεντρικό μικροελεγκτή. Έχει εύρος τιμών από τους - 55 έως +125 °C και ακρίβεια ±0.5 °C όταν βρίσκεται ανάμεσα στους -10 έως +85 °C. Μπορεί, επιπλέον, να λειτουργήσει παρασιτικά στο κύκλωμα, τροφοδοτούμενο από την γραμμή δεδομένων, με αποτέλεσμα να μην χρειαστεί εξωτερική τροφοδοσία. [13]

3.2.2 Ηλεκτρονόμος (Relay)

Τα GPIO pins του Raspberry Pi και άλλων μικροϋπολογιστών (Arduino, Beaglebone κλπ) επιτρέπουν την κατασκευή κυκλωμάτων και εφαρμογών που χρειάζονται το πολύ 3-5 Volt. Είναι απλά, δηλαδή κυκλώματα που περιλαμβάνουν LEDS, απλούς αισθητήρες επαφής, θερμοκρασίας και γενικά εφαρμογές

Σχήμα 3.3: Αισθητήρας θερμοκρασίας DS18B20

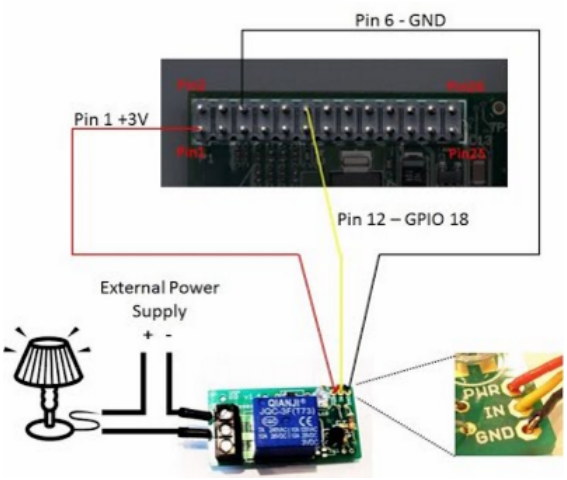
περισσότερο εκπαιδευτικού σκοπού. Για εφαρμογές που περιλαμβάνουν συσκευές οι οποίες χρειάζονται πιο ψηλή τάση, 220-240 Volts, χρειάζεται ένα εξάρτημα το οποίο θα διαβάζει το σήμα από τα pins του μικροεπεξεργαστή και θα τις τροφοδοτεί με το κατάλληλο ηλεκτρικό ρεύμα. Ένα τέτοιο εξάρτημα είναι το Relay Module 250V/10A στην εικόνα 3.4. Το relay επίσης προστατεύει το μικροϋπολογιστή μας από το ψηλό οικιακό ρεύμα.

Σχήμα 3.4: Relay για raspberry pi ή arduino

Το relay χρησιμεύει για έλεγχο διακόπτη φωτιστικού, θερμοσίφωνα και οποιασδήποτε άλλης ON/OFF συσκευής. Για παράδειγμα ο έλεγχος ενός διακόπτη φωτιστικού μπορεί να γίνει υλοποιώντας την διάταξη στην εικόνα 3.5. Υπάρχει ψηλός κίνδυνος ατυχήματος όταν υλοποιεί κανείς εφαρμογές με ψηλή τάση, γι' αυτό συστήνεται η σύνδεση στο ηλεκτρικό δίκτυο, να γίνεται από κάποιο ειδικό.

3.3 Apache HTTP Server

Ο Apache HTTP Server, συχνά αναφερόμενος απλά ως Apache, είναι το πιο επιτυχυμένο λογισμικό εξυπηρέτησης Ιστού. Το 2009 έγινε το πρώτο λογισμικό web server, καταφέρνοντας να ξεπεράσει τους εκατό εκατομμύρια



Σχήμα 3.5: Σχεδιάγραμμα διακόπτη relay ελεγχόμενο από raspberry pi

ιστοτόπους. Ο Apache ήταν η πρώτη βιώσιμη εναλλακτική λύση για το web server της εταιρείας Netscape Communications (σήμερα γνωστός ως Sun Java System), και από τότε έχει εξελιχθεί σε άξιο ανταγωνιστή άλλων web server που βασίζονται σε περιβάλλον Unix, όσον αφορά την λειτουργικότητα και την απόδοση.

Ο Apache αναπτύσσεται και συντηρείται από μια ανοιχτή κοινότητα προγραμματιστών στο πλαίσιο του Apache Software Foundation. Η εφαρμογή είναι διαθέσιμη για μια ευρεία ποικιλία λειτουργικών συστημάτων, συμπεριλαμβανομένων των Unix, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, και πολλά άλλα. Εκδίδεται σύμφωνα με την άδεια χρήσης Apache και χαρακτηρίζεται ως λογισμικό ανοικτού κώδικα.

Ο Apache υποστηρίζει μια ποικιλία χαρακτηριστικών, πολλά από τα οποία εφαρμόζονται υπό την μορφή modules και επεκτείνουν τη λειτουργικότητα του πυρήνα. Αυτά μπορεί να κυμαίνονται από serverside γλώσσες προγραμματισμού, μέχρι υποστήριξη για συστήματα ελέγχου ταυτότητας. Υποστηρίζει γλώσσες προγραμματισμού όπως PHP, Perl, Python και Tcl. Μερικά από τα πιο δημοφιλή modules είναι τα: mod_access, mod_auth, mod_digest, και mod_auth_digest. Ένα δείγμα από άλλα χαρακτηριστικά γνωρίσματα περιλαμβάνουν SSL και TLS υποστήριξη (mod_ssl), επανεγγραφή URL (mod_rewrite), προσαρμοσμένα αρχεία καταγραφής (mod_log_config) και υποστήριξη φιλτραρίσματος (mod_include και mod_ext_filter).

Ένα ακόμα χαρακτηριστικό του Apache είναι η δυνατότητα Virtual Hosting η οποία επιτρέπει σε μία εγκατάσταση Apache να εξυπηρετεί πολλές διαφορετικές πραγματικές ιστοσελίδες. Για παράδειγμα, ένας server με μία μόνο εγκατάσταση Apache θα μπορούσε να εξυπηρετεί ταυτόχρονα τους ιστότοπους www.example.com, www.test.com, test47.test-server.test.com, κλπ. Τέλος ο Apache διαθέτει την δυνατότητα ρύθμισης μηνυμάτων λάθους, υποστηρίζει την δυνατότητα διασύνδεσης με Συστήματα Διαχείρισης Βάσεων Δεδομένων (DBMS) και υποστηρίζεται από πολλές γραφικές διεπαφές χρήστη (GUI). [14]

3.4 Βάση δεδομένων MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων ανοικτού κώδικα (relational database management system - RDBMS), που χρησιμοποιεί την Structured Query Language (SQL). Την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων. Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να κατεβάσει τη MySQL και να την διαμορφώσει με βάση τις ανάγκες του, σύμφωνα πάντα με την γενική άδεια χρήσης. Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει. Οι περισσότεροι συμφωνούν ωστόσο ότι δουλεύει καλύτερα όταν διαχειρίζεται περιεχόμενο και όχι όταν εκτελεί συναλλαγές. Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.

Η MySQL μετρά περισσότερες από 11 εκατομμύρια εγκαταστάσεις. Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Είναι η πιο δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες. Χρησιμοποιείται σε κάποιες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες, όπως το Flickr,το YouTube,η Wikipedia,το Google,το Facebook και το Twitter. [14]

3.5 Eclipse Paho

Το έργο Eclipse PAHO παρέχει ανοιχτού-κώδικα εφαρμογές πελάτη MQTT και πρωτόκολλα ανταλλαγής μηνυμάτων MQTT - SN με στόχο τις νέες, τις υφιστάμενες και τις αναδυόμενες εφαρμογές για Machine-to-Machine (M2M) και το Διαδίκτυο των πραγμάτων (IoT). Είναι μέρος της αποστολής M2M του Ιδρύματος Eclipse (Eclipse Foundation's M2M) για την παροχή υψηλής ποιότητας εφαρμογών των βιβλιοθηκών και εργαλείων M2M. Με την ονομασία PAHO, επιμελούνται και αναπτύσσονται ανοιχτού-κώδικα βιβλιοθήκες πελάτη για MQTT (MQTT client). Υπάρχουν ήδη MQTT C και Java βιβλιοθήκες με Lua , Python , C ++ και JavaScript σε διάφορα στάδια της ανάπτυξης.

Κάτω από το ίδρυμα Eclipse υπάρχει και ο Mosquitto broker. Παρά το γεγονός ότι πολλοί MQTT brokers είναι διαθέσιμοι, ο Mosquitto είναι μακράν ο ευκολότερος να ρυθμιστεί και να τρέξει για μια MQTT εφαρμογή. Επίσης είναι ανοικτή πηγή, έτσι μπορεί ο καθένας να το κατεβάσει και να το εκτελέσει στο δικό του σύστημα, είτε πρόκειται για Windows, Mac OS X, Linux ή πολλές άλλες πλατφόρμες.[16]

3.6 Λογιμικό Android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας

προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής υλικού (hardware), οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Ιρίνα Μπλόκ.

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί. [17]

Στη συνέχεια επεξηγούνται μερικές από τις πιο βασικές ορολογίες που συναντά κανείς όταν υλοποιεί μια εφαρμογή android:

- **Activity**
  Ένα Activity είναι μια διεπαφή χρήστη που επιτρέπει στο χρήστη να αλληλεπιδρά με την οθόνη, για την εκτέλεση ενεργειών. Όταν ξεκινήσει μια εφαρμογή, αυτό που εμφανίζει είναι το αποτέλεσμα ενός Activity. Σε επίπεδο κώδικα, για δημιουργηθεί μια δραστηριότητα , θα πρέπει να δημιουργηθεί μία κλάση που επεκτείνει την κλάση Activity . Μια δραστηριότητα έχει μια απαιτούμενη onCreate μέθοδο. Είναι η κύρια μέθοδος . Για να αλληλεπιδράσει το πρόγραμμα και τα Activities, πρέπει να υπάρχει κάτι που εμφανίζεται, γι' αυτό και τα Activities περιέχουν αυτό που ονομάζεται views.

- **View**
  Το View είναι το βασικό δομικό στοιχείο για στοιχεία διεπαφής χρήστη. Ένα View καταλαμβάνει μια ορθογώνια περιοχή στην οθόνη. Το View είναι η βασική κλάση για τα widgets, τα οποία χρησιμοποιούνται για να δημιουργήσουν διαδραστικά στοιχεία UI (Κουμπιά , πεδία κειμένου κλπ). Υπάρχουν διαφορετικά είδη Views, για παράδειγμα ένα ListView είναι σε θέση να εμφανίζει μια διαδραστική λίστα, ενώ ένα WebView επιτρέπει να εμφανιστεί μια ιστοσελίδα. Για την οργάνωση αυτών των ορθογωνίων στην οθόνη, υπάρχει είναι ένα αρχείο κειμένου γραμμένο σε XML για κάθε διαφορετική οθόνη.

- **Xml**
  Xml σημαίνει Extensible Markup Language . Το λογισμικό Android παρέχει ένα απλό λεξιλόγιο XML που αντιστοιχεί στις View κατηγορίες και υποκατηγορίες. Ο στόχος της χρήσης του XML λεξιλογίου του Android, είναι για γρήγορο σχεδιασμό του UI και των στοιχείων της οθόνης που περιέχει.

- **Service**
  Ένα Service είναι ένα στοιχείο της εφαρμογή που μπορεί να εκτελέσει μακροχρόνια εργασίες στο παρασκήνιο(background) και δεν παρέχει περιβάλλον εργασίας χρήστη. Ένα άλλο στοιχείο της εφαρμογής μπορεί να ξεκινήσει ένα Service και θα συνεχίσει να εκτελείται στο παρασκήνιο, ακόμη και αν ο χρήστης μεταβαίνει σε μια άλλη εφαρμογή. Επιπλέον, ένα άλλο συστατικό μπορεί να συνδεθεί σε ένα Service για να αλληλεπιδρούν και να έχουν ενδοεπικοινωνία (IPC). Για παράδειγμα, ένα Service μπορεί να χειριστεί τις συναλλαγές δικτύου, να παίζει μουσική, να χειριστεί ένα αρχείο I/O, ή να αλληλεπιδράσει με το UI. Όλα αυτά στο παρασκήνιο.

Κεφάλαιο 4

MQTT

4.1 Εισαγωγή στο MQTT

Το MQTT είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων πάνω από TCP/IP, με προεπιλεγμένη θύρα τη 1883. Είναι ελαφρύ, απλό και εύχρηστο στην εφαρμογή του, χαρακτηριστικά που το κάνουν ιδανικό για M2M επικοινωνίες και IoT εφαρμογές όπου απαιτείται περιορισμένος όγκος κώδικα και το εύρος ζώνης του δικτύου είναι περιορισμένο. Ακόμη η χαμηλή κατανάλωση ενέργειας είναι άλλος ένας λόγος που συχνά επιλέγεται και για εφαρμογές κινητού. Στην συνέχεια γίνεται μια λεπτομερής ανάλυση του τρόπου λειτουργίας αφού το MQTT είναι ο κορμός των επικοινωνιών στην πλατφόρμα που αναπτύξαμε. [9] [20]

4.2 Ιστορικά

Το MQTT εφευρέθηκε από τους Dr.Andy Stanford-Clark της IBM και Arlen Nipper της Arcom (τώρα Eurotech) το 1999. Ο στόχος τους ήταν να δημιουργήσουν ένα πρωτόκολλο για παρακολούθηση σωληνώσεων πετρελαίου μέσω δορυφορικής σύνδεσης το οποίο θα είχε ελάχιστη ανάγκη από μπαταρίες και εύρος ζώνης (bandwidth). Ξεκινώντας την σχεδίασή του είχαν τις εξής προδιαγραφές στο μυαλό τους:

- Απλό στην εφαρμογή.
- Να παρέχει Quality of Service (QoS).
- Αποδοτικό ως προς την κατανάλωση ενέργειας και εύρους ζώνης
- Λειτουργικό ανεξαρτήτως δεδομένων (που αποστέλλονται)
- Συνεχής σύνδεση

Αυτές οι προδιαγραφές είναι ακόμη ο πυρήνας του MQTT παρόλο που ο προσανατολισμός άλλαξε από ενσωματωμένα συστήματα σε IoT εφαρμογές.

Ένα θέμα για το οποίο γίνεται συζήτηση είναι για το τι σημαίνει το ακρωνύμιο MQTT. Η σύντομη απάντηση είναι ότι δεν έχει ακρωνύμιο πλέον, το πρωτόκολλο ονομάζεται απλά MQTT. Εκτενέστερα, ως ακρωνύμιο υπήρχε το MQ Telemetry Transport όπου το MQ αναφέρεται σε ένα προϊόν της IBM (International Business Machines), το MQ Series, το οποίο υποστήριζε το MQTT και από εκεί πήρε και το όνομα του το πρωτόκολλο. Είναι λάθος να ονομάζεται πρωτόκολλο σειριακών μηνυμάτων (Message Queue protocol) γιατί δεν είναι σειριακό πρωτόκολλο με τη παραδοσιακή έννοια παρόλο που υποστηρίζει σειριοποίηση όπως θα δούμε στη συνέχεια.

Τελικά, η IBM μετά από χρήση του MQTT σε αρκετές δικές τις εφαρμογές έβγαλε το 2010 την έκδοση 3.1 δωρεάν και έτσι ο καθένας μπορεί να το χρησιμοποιήσει και να υλοποιήσει εφαρμογές. Από τότε αρκετές εταιρίες και οργανισμοί έχουν φτιάξει ποικίλες εφαρμογές στηριζόμενες σε αυτό, άλλες ανοιχτού-κώδικα και άλλες κλειστού-κώδικα, κάποιες κερδοσκοπικά κάποιες όχι. Ποιο κάτω είναι μερικές από τις μεγαλύτερες υλοποιήσεις βασισμένες στο MQTT πρωτόκολλο:

- Facebook Messenger
- WarmDirt - Ένα έργο που ελέγχει τη θερμοκρασία στο χώμα, για φυτά
- homA - Framework για αυτοματισμό σπιτιού
- Relayr - Μία IoT εταιρία που χρησιμοποιεί το MQTT για να συνδέσει IoT συσκευές.

Περισσότερες εφαρμογές υπάρχουν σε λίστα στο link : https://github.com/-mqtt/mqtt.github.io/wiki/Example

4.3 Publish/Subscribe Μοντέλο

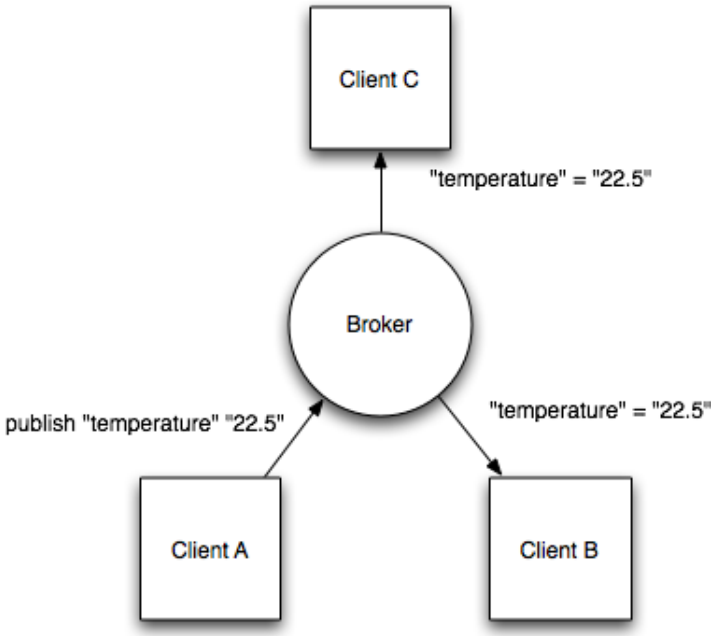
Το publish/subscribe (pub/sub) μοντέλο έρχεται ως εναλλακτική στο παραδοσιακό client-server μοντέλο όπου 2 συσκευές επικοινωνούν απευθείας. Εδώ ο client που στέλνει ένα μήνυμα (pub) δεν ξέρει την ύπαρξη του παραλήπτη-client (sub). Υπάρχει ένα 3^o κομμάτι του παζλ, ο broker, ο οποίος είναι γνωστός τόσο στον αποστολέα (publisher) όσο και στον παραλήπτη (subscriber). Η δουλειά του broker είναι να λαμβάνει όλα τα εισερχόμενα μηνύματα και να τα προωθεί κατάλληλα ώστε συγκεκριμένα μηνύματα να φτάνουν σε συγκεκριμένους παραλήπτες.

Ας δούμε λίγο καλύτερα τα πιο πάνω μέσω ενός παραδείγματος. Στην εικόνα 4.1 ο client A θέλει να ενημερώσει την τιμή της θερμοκρασίας. Δεν ξέρει όμως ποιος/οι είναι αυτοί που θα παραλάβουν την πληροφορία. Στέλνει την τιμή στον Broker και αυτός την προωθεί στον client B και C που τους ενδιαφέρει.

Ουσιαστικά με αυτό το μοντέλο πετυχαίνουμε 3 είδη αποσύνδεσης.

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

Ο τρόπος που αποστέλνονται και φιλτράρονταν τα μηνύματα είναι ο λόγος που επιτρέπει να υπάρχει αυτή η αποσύνδεση. Όταν μια συσκευή στέλνει ένα μήνυμα το συσχετίζει με ένα θέμα (topic). Τα topics είναι συμβολοσειρές σε ιεραρχική δομή χωρισμένες με το χαρακτήρα '/'. Με αυτό το τρόπο το MQTT μας δίνει ευελιξία και καλύτερη οργάνωση. Για παράδειγμα έστω ότι ο Client A στην εικόνα 4.1 βρίσκεται στο σαλόνι. Μια καλή τακτική είναι να στείλει τη θερμοκρασία στο topic livingroom/temperature, και όποιος θέλει να ξέρει αυτή τη θερμοκρασία πρέπει να κάνει subscribe στο αντίστοιχο topic. Αυτή η δομή μας δίνει επίσης το πλεονέκτημα να κάνουμε χρήση κάποιων wildcards που υπάρχουν. Ο χαρακτήρας '#' ως τελευταίο κομμάτι του topic, επιτρέπει να κάνουμε subscribe σε όλο το υποδέντρο. Πιο συγκεκριμένα με subscribe στο topic livingroom/ θα λαμβάνονται τα δεδομένα τόσο του αισθητήρα θερμοκρασία, αλλά και όποιας άλλης συσκευής-αισθητήρα στέλνει δεδομένα κάτω από το livingroom/, π.χ livingroom/humidity, livingroom/light_switch κλπ. Το άλλο σημαντικό wild- card είναι ο χαρακτήρας '+'. Στην θέση του '+' θα αποκατασταθεί οτιδήποτε είναι γνωστό στο Broker ώστε να συμπληρω-



Σχήμα 4.1: Publish/Subscribe Μοντέλο

θεί κάποιο topic. Για παράδειγμα το home/+/temp θα ταιριάξει και με το home/bedroom/temp και με το home/livingroom/temp.

Ένα μεγάλο πλεονέκτημα του Pub/Sub μοντέλου έναντι του παραδοσιακού client-server είναι ότι οι διαδικασίες στο broker είναι υψηλά παραλληλοποιήσιμες και έτσι επιτρέπει στις εφαρμογές μας να κλιμακώνουν αρκετά.

Ένα μειονέκτημα είναι ότι τόσο ο publisher όσο και ο subscriber πρέπει να είναι γνώστες της δομής των topics.

4.4 Client, Broker και η σύνδεση τους

4.4.1 Client

Ο όρος MQTT client συμπεριλαμβάνει τόσο τον publisher όσο και τον subscriber. Στην γενική περίπτωση ένας MQTT client μπορεί να είναι και τα δύο. Ένας client μπορεί να είναι από ένας μικρο-ελεγκτής μέχρι ένα πλήρες υπολογιστικό σύστημα το οποίο έχει μια MQTT βιβλιοθήκη και είναι συνδεδεμένος σε ένα MQTT broker πάνω από οποιοδήποτε δίκτυο. Η απήχηση του MQTT μπορεί να γίνει κατανοητή και από την ποικιλία σε βιβλιοθήκες για MQTT client που μπορεί να βρει κανείς, σε Arduino, Android, Java , JavaScript, .NET, C, C++ και πολλά άλλα. [10]

4.4.2 Broker

Η καρδιά οποιουδήποτε pub/sub μοντέλου είναι ο broker. Ανάλογα με την υλοποίηση ο broker μπορεί να διαχειριστεί ταυτόχρονα χιλιάδες συνδέσεις. Η πρωταρχική του δουλειά είναι να λαμβάνει, να φιλτράρει και να προωθεί σωστά τα μηνύματα. Επιπρόσθετα κρατά όλα τα δεδομένα που αφορούν τους clients οι οποίοι έχουν κάνει "επίμονη" σύνδεση (persisted clients). Πιο κάτω παρατίθενται περισσότερες πληροφορίες γι' αυτό. Ευθύνη του broker είναι επίσης και ο έλεγχος για εξουσιοδοτημένους clients αν έχει προστεθεί τέτοια υλοποίηση.

4.4.3 Η σύνδεση

Αρχικά τόσο ο client όσο και ο broker για να τρέξουν χρειάζονται μια στοίβα TCP/IP. Η MQTT σύνδεση γίνεται πάντα από ένα client και το broker, ποτέ client με client. Η σύνδεση αρχικοποιείται με το client να στέλνει ένα CONNECT μήνυμα και ο broker να απαντά με ένα CONNACK και τη κατάσταση σύνδεσης. Όταν η σύνδεση εγκατασταθεί ο broker θα τη διατηρήσει μέχρι ο client να στείλει εντολή αποσύνδεσης ή να χάσει τη σύνδεση του (από το δίκτυο). Ένα CONNECT μήνυμα μεταξύ άλλων περιλαμβάνει:

- clientId - Ένα μοναδικό αναγνωριστικό για κάθε MQTT client. Αν δεν οριστεί από το client ο broker θα του δώσει ένα τυχαίο αριθμό.

- Clean Session - Αυτή η μεταβλητή δηλώνει κατά πόσο ο client θέλει μόνιμη σύνδεση (persistent session) ή όχι. Τέτοια σύνδεση (Clean Session=false) σημαίνει ότι ο broker θα αποθηκεύσει όλα τα subscriptions και όλα τα χαμένα μηνύματα (με QoS>0) που αφορούσαν αυτό τον client ενώ ήταν αποσυνδεδεμένος.

- Username/Password - Αν ο broker ρυθμιστεί κατάλληλα ο client πρέπει να πιστοποιήσει την αυθεντικότητα του.

- Will Message - Έχουμε την δυνατότητα κατά την σύνδεση να ρυθμίσουμε ένα μήνυμα και όταν ο client αποσυνδεθεί αναπάντεχα τότε ο broker θα αναλάβει να το προωθήσει κατάλληλα στην θέση του client σαν τελευταίο μήνυμα.

- KeepAlive - Είναι το χρονικό περιθώριο στο οποίο δεσμεύεται ο client να στέλνει ένα PING μήνυμα και ο broker να στέλνει μια PING απάντηση ώστε και οι 2 πλευρές να ξέρουν ότι η σύνδεση είναι ενεργή.

4.5 Publish και Subscribe σε λεπτομέρειες

4.5.1 Publish

Όπως ήδη αναφέρθηκε ένας client στέλνει μηνύματα σε συγκεκριμένα topics. Άρα μια εντολή publish αναμφίβολα πρέπει να περιέχει το topic και το μήνυμα. Το MQTT όμως είναι data-agnostic δηλαδή δεν ασχολείται με το περιεχόμενο και τη δομή του μηνύματος, είναι δουλειά του χρήστη να αποφασίσει αν θέλει δυαδικά δεδομένα, απλό κείμενο ή σε κάποιο τύπο όπως JSON και XML. Πιο συγκεκριμένα μια εντολή publish περιέχει τουλάχιστον τα παρακάτω:

- Topic Name - Μια συμβολοσειρά ιεραρχικά δομημένη με το χαρακτήρα '/'.

- Quality of Service (QoS) - Μπορεί να είναι 0,1 ή 2. Κάθε επίπεδο παρέχει διαφορετική εγγύηση για την αποστολή του μηνύματος. Λεπτομέρειες πιο κάτω.

- Retain Flag - Αν η τιμή αυτής της μεταβλητής είναι αληθής το μήνυμα θα αποθηκευτεί από το broker ως τελευταία γνωστή τιμή και νέοι subscribers θα λαμβάνουν αυτό το μήνυμα αμέσως μετά τη σύνδεσή τους.

- Payload - Εδώ αποθηκεύεται το μήνυμα σε όποια δομή επιθυμεί ο χρήστης.

- Packet Identifier - Μοναδικό αναγνωριστικό μεταξύ broker και client ώστε να ξέρουν την πορεία του μηνύματος. Έχει νόημα για QoS>0. Επίσης καθορίζεται αυτόματα από την αντίστοιχη MQTT βιβλιοθήκη.

- DUP flag - Καθορίζει αν αυτό το μήνυμα είναι αντίγραφο κάποιου άλλου και ξαναστέλνεται επειδή το προηγούμενο δεν έφτασε στον παραλήπτη. Έχει νόημα για QoS>0. Είναι μέρος ενός εσωτερικού μηχανισμού για αξιόπιστη αποστολή-παραλαβή των μηνυμάτων.

4.5.2 Subscribe

Η αποστολή μηνυμάτων δεν έχει νόημα αν δεν υπάρχει κάποιος να τα παραλάβει. Ένας client χρειάζεται να εκτελέσει την εντολή subscribe για να λάβει τα μηνύματα. Η εντολή subscribe είναι απλή και περιλαμβάνει τα ακόλουθα:

- Packet Identifier - Μοναδικό αναγνωριστικό μεταξύ broker και client ώστε να ξέρουν την πορεία του μηνύματος. Αντίστοιχο με αυτό στην publish εντολή.
- Λίστα από Subscriptions - Κάθε subcription είναι ένα ζεύγος από topic και επίπεδο QoS. Αν υπάρχουν πολλαπλά subscriptions για το ίδιο topic λαμβάνεται υπόψη αυτό με το μεγαλύτερο QoS.

Κάθε subscription θα επικυρωθεί από το broker στέλνοντας πίσω ένα SUBACK μήνυμα.

4.5.3 Unsubscribe

Αν κάποιος client δεν επιθυμεί να λαμβάνει πλέον μηνύματα από κάποιο topic μπορεί να διαγράψει το subscription. Η εντολή unsubscribe έχει την ίδια δομή με την subscribe. Επίσης και σε αυτή τη περίπτωση ο broker θα απαντήσει με ένα UNSUBACK μήνυμα ώστε να επικυρώσει το unsubscribe.

4.6 Επίπεδα QoS (Quality of Service)

Έχουμε ήδη συναντήσει τον όρο Quality of Service και τώρα ήρθε η ώρα να τον αναλύσουμε εις βάθος.

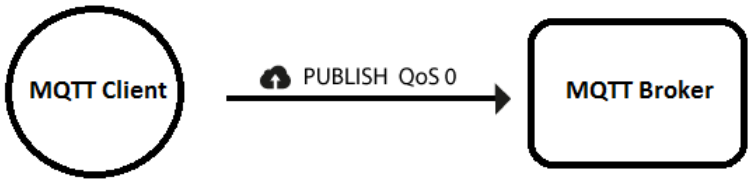
Το επίπεδο QoS είναι μια συμφωνία μεταξύ αποστολέα και παραλήπτη για το πόσο εγγυημένη είναι η παράδοση ενός μηνύματος. Όσο αφορά το QoS η διαδρομή του μηνύματος από τον publisher στο subscriber μπορεί να εξεταστεί από 2 πλευρές. Κατά πρώτον από το Qos που αφορά την αποστολή μηνύματος από τον Publisher στον broker, όπως έχει καθοριστεί από την εντολή publish. Κατά δεύτερο από το Qos για την αποστολή μηνύματος από τον broker στο subscriber, όπως έχει καθοριστεί από την εντολή subscribe. Συνεπώς, το QoS μπορεί να υποβιβαστεί για κάποιο παραλήπτη ο οποίος έκανε subscribe με μικρότερο QoS. Στο MQTT υπάρχουν 3 QoS επίπεδα:

- QoS 0 - Το πολύ μια φορά. Το ελάχιστο επίπεδο QoS, το μήνυμα αποστέλλεται μια φορά και δεν λαμβάνεται επιβεβαίωση παράδοσης ούτε αποθηκεύεται. Συνήθως είναι όσο αξιόπιστο είναι και το TCP πρωτόκολλο πάνω από το οποίο γίνεται η αποστολή. Αυτό το επίπεδο επιλέγεται όταν υπάρχει σταθερή σύνδεση broker και client (συνήθως ενσύρματη) ή όταν δεν υπάρχει πρόβλημα να χάσουμε λίγη πληροφορία.
- QoS 1 - Τουλάχιστον, μια φορά. Ο αποστολέας αποθηκεύει το μήνυμα μετά την αποστολή του μέχρι να πάρει το PUBACK μήνυμα. Αν περάσει συγκεκριμένο χρονικό διάστημα και δεν το λάβει προσπαθεί να αποστείλει το μήνυμα ξανά. Με αυτό το τρόπο το μήνυμα μπορεί να φτάσει

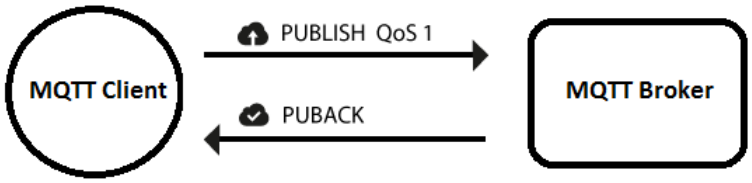
περισσότερες φορές. Αυτό το επίπεδο επιλέγεται όταν στόχος είναι να φτάσει σίγουρα η πληροφορία και εναπόκειται στο χρήστη αν θα χειριστεί την επιπρόσθετη πληροφορία (τα αντίγραφα).

- QoS 2 - Σίγουρα μια φορά. Είναι ο ασφαλέστερος, αλλά και ο πιο αργός τρόπος να σταλεί ένα μήνυμα. Για να συμβεί αυτό γίνονται 2 ανταλλαγές μηνυμάτων. Αν ο client στείλει δεύτερη φορά ένα μήνυμα, γιατί καθυστέρησε η απάντηση ο broker θα προωθήσει μόνο το ένα. Αυτό το επίπεδο επιλέγεται όταν είναι πολύ σημαντικό για την εφαρμογή να λαμβάνονται όλα τα μηνύματα ακριβώς μια φορά και η επιβάρυνση από το διπλό έλεγχο δεν μας είναι τόσο μειονέκτημα.

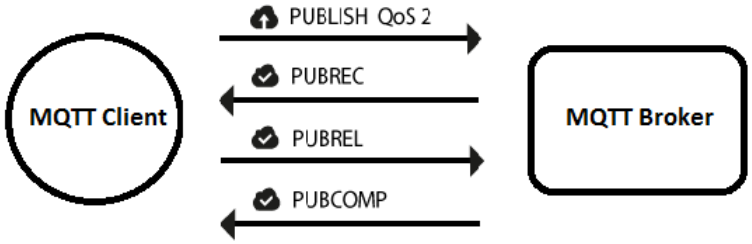
Το QoS είναι σημαντικό χαρακτηριστικό του πρωτοκόλλου MQTT. Με τον έλεγχο της παράδοσης του μηνύματος να είναι στα χέρια του MQTT μπορεί να κάνει την επικοινωνία σε αναξιόπιστα δίκτυα ευκολότερη. Επιπλέον δίνει την ευχέρεια στο χρήστη να διαλέξει το QoS επίπεδο ανάλογα με το είδος της εφαρμογής του.



Σχήμα 4.2: QoS 0



Σχήμα 4.3: QoS 1



Σχήμα 4.4: QoS 2

4.7 Επίμονη σύνδεση και σειριοποίηση μηνυμάτων

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

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

Στο κλασσικό πρωτόκολλο σειριακών μηνυμάτων τα μηνύματα σειριοποιούνται μέχρι κάποιος καταναλωτής-client να τα παραλάβει. Σε αντίθετη περίπτωση θα παραμείνουν "κολλημένα" στη σειρά περιμένοντας να καταναλωθούν. Δεν είναι δυνατό κάποιο μήνυμα να μην μπορεί να σταλεί σε κάποιο client όπως συμβαίνει στο MQTT αν δεν κάνει κανείς subscribe σε αυτό το topic. Επιπλέον, μεγάλη διαφορά είναι ότι στο κλασσικό πρωτόκολλο σειριακών μηνυμάτων κάθε μήνυμα έχει μονάχα ένα παραλήπτη ενώ όπως είδαμε στο MQTT παραλήπτες μπορεί να είναι όσοι έχουν κάνει subscribe στο αντίστοιχο topic.

4.8 Διατηρημένα μηνύματα (Retained Messages)

Ένα μήνυμα ονομάζεται διατηρημένο όταν κατά την αποστολή του η μεταβλητή retainFlag τεθεί ως αληθής. Σε αυτή τη περίπτωση ο broker θα αποθηκεύσει το μήνυμα καθώς και το QoS επίπεδο. Ακολούθως όποιος client εκτελέσει την εντολή subscribe σε topic που ταιριάζει με το topic στο διατηρημένο μήνυμα θα λάβει αυτό το μήνυμα ως την τελευταία γνωστή τιμή. Για κάθε topic μόνο 1 διατηρημένο μήνυμα μπορεί να υπάρχει. Αυτό που ήρθε τελευταίο αντικαθιστά το προηγούμενο.

Είναι σημαντικό να γίνει αντιληπτό ότι τα διατηρημένα μηνύματα δεν έχουν καμία σχέση με την επίμονη σύνδεση και την αποθήκευση μηνυμάτων, τα οποία αναφέρθηκαν πιο πάνω πιο πάνω.

Έχει λεχθεί πώς στέλνεται ένα διατηρημένο μήνυμα. Για να διαγραφεί πρέπει απλώς να σταλεί ένα καινούργιο με 0 byte μέγεθος.

Η μεγαλύτερη χρησιμότητα των διατηρημένων μηνυμάτων είναι να μην κρατιούνται στο σκοτάδι νέοι subscribers. Για παράδειγμα ένας νέος subscriber στο topic myhome/devices/device1/status θα λάβει την τελευταία γνωστή κατάσταση (online/offline) της συσκευής χωρίς να χρειάζεται να περιμένει να γίνει η επόμενη μέτρηση.

4.9 Ασφαλής σύνδεση με το MQTT

Στο κεφάλαιο 2 έχει παρουσιαστεί το θέμα για την ασφάλεια σε IoT εφαρμογές. Σε αυτή την παράγραφο θα αναλύσουμε τι επιλογές ασφαλείας που προσφέρει το MQTT.

4.9.1 Πιστοποίηση

Πιστοποίηση (authentication) είναι η διαδικασία που ακολουθείται για να είναι βέβαιο ότι ένα πρόσωπο, μια συσκευή ή μια εφαρμογή έχει την ταυτότητα που δηλώνει. Για παράδειγμα στο αεροδρόμιο και γενικά στην καθημερινή μας ζωή όταν χρειάζεται να δείξουμε ταυτότητα ή διαβατήριο είναι μια διαδικασία πιστοποίησης. Ακόμη και ο κωδικός στον υπολογιστή μας εμπίπτει σε αυτή την κατηγορία.

Το MQTT πρωτόκολλο παρέχει την επιλογή για πιστοποίηση κατά την σύνδεση στον broker (πεδία username,password). Για να έχει νόημα η αποστολή usename και password στον broker, ο τελευταίος πρέπει να ρυθμιστεί κατάλληλα ώστε να επαληθευτούν αυτά τα στοιχεία κατά τη σύνδεση δίνοντας του από πριν σε συγκεκριμένο αρχείο τους εγκεκριμένους χρήστες. Προσοχή, τα στοιχεία στέλνονται σε απλό κείμενο. Με οποιαδήποτε κλοπή των δεδομένων κατά την μεταφορά τους θα μπορούν να διαβαστούν και να χρησιμοποιηθούν άμεσα. Για να αποτραπεί αυτό, τα δεδομένα μπορούν να κρυπτογραφηθούν.

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

4.9.2 Εξουσιοδότηση

Ο όρος αυτός αφορά το κατά πόσο ένα πρόσωπο, μια συσκευή ή μια εφαρμογή έχει δικαίωμα σε συγκεκριμένες λειτουργίες και δεδομένα. Για παράδειγμα ενώ το διαβατήριο θα πιστοποιήσει την ταυτότητα μας στο αεροδρόμιο αυτό που θα μας δώσει πρόσβαση σε συγκεκριμένο αεροπλάνο είναι το εισιτήριο. Γενικά η πιστοποίηση και η εξουσιοδότηση είναι 2 "καλοί φίλοι". Δεν έχει νόημα να επιτραπεί μια ενέργεια αν δεν έχει προηγηθεί έλεγχος της ταυτότητας.

Σε μεγάλα συστήματα επικοινωνίας τα οποία χρησιμοποιούν το MQTT πρωτόκολλο η εξουσιοδότηση μπορεί να φανεί πολύ χρήσιμη. Η εξουσιοδότηση, που δίνεται στο client αφορά τα δικαιώματα του στα topics και πρέπει να μπορεί να ρυθμιστεί, ενώ ο broker τρέχει (runtime), για ευνόητους λόγους. Μια εξουσιοδότηση σε ένα topic μπορεί να συμπεριλαμβάνει τα εξής:

- Το ακριβές topic ή κάποιο το οποίο συμπεριλαμβάνει wildcards.
- Δικαίωμα ή όχι για την εκτέλεση των εντολών publish και/ή subscribe.
- Το επίπεδο QoS που έχει δικαίωμα να χρησιμοποιήσει (0,1,2 ή όλα).

4.9.3 Κρυπτογράφηση

Ας πάρουμε ένα σενάριο στο οποίο στέλνουμε ένα γράμμα μέσω ταχυδρομείου. Έχουμε καθορίσει τον παραλήπτη και ο ταχυδρόμος θα φροντίσει να φτάσει στον προορισμό του. Κανείς όμως δεν μας εγγυάται, ότι ο ταχυδρόμος ή όποιος άλλος εμπλέκεται στην μεταφορά δεν θα διαβάσει το γράμμα ή ακόμη χειρότερα να αλλάξει το περιεχόμενο του.

Το ίδιο σενάριο ισχύει και όταν στέλνονται δεδομένα σε απλό κείμενο μέσω ενός δικτύου γενικά και πιο συγκεκριμένα στο διαδίκτυο. Τα TCP πακέτα θα περάσουν από διάφορες υποδομές (routers, firewalls, Internet Exchange Points κλπ) πριν φτάσουν στο προορισμό τους. Το πακέτο με τα δεδομένα μπορεί να διαβαστεί και να τύχει κακόβουλης επεξεργασίας από κάθε σημείο.

Για αποφυγή του πιο πάνω κίνδυνου υπάρχουν τα πρωτόκολλα κρυπτογραφίας TLS (Transport Layer Security) και SSL (Secure Sockets Layer). Τα πρωτόκολλα TLS/SSL χρησιμοποιούν μηχανισμούς χειραψίας ώστε να ρυθμίσουν κατάλληλες παραμέτρους για να δημιουργήσουν μια ασφαλή σύνδεση ανάμεσα σε client και server. Σε αυτή την σύνδεση πλέον τα δεδομένα μεταφέρονται κρυπτογραφημένα και με την παρούσα τεχνολογία θεωρείται αδύνατο να διαβαστούν.

Η πιστοποίηση και εξουσιοδότηση είναι πρακτικές ασφαλείας σε επίπεδο εφαρμογής. Σε επίπεδο μεταφοράς δεδομένων στο MQTT υπάρχει η δυνατότητα TLS σύνδεσης αντί απλής TCP, στη θύρα 8883. Σαφώς και αυτή η σύνδεση επιφέρει επιβάρυνση σε υπολογιστικούς πόρους, αλλά συστήνεται ακράδαντα ειδικά όταν γίνεται χρήση username και password. Στην περίπτωση που η συσκευή δεν μπορεί να στηρίξει TLS σύνδεση, εναλλακτική μέθοδος είναι να κρυπτογραφηθούν τόσο τα μηνύματα (το πεδίο payload στην εντολή publish) όσο και ο κωδικός (στην εντολή connect) σε επίπεδο εφαρμογής.

Κεφάλαιο 5

Ανάλυση Πλατφόρμας

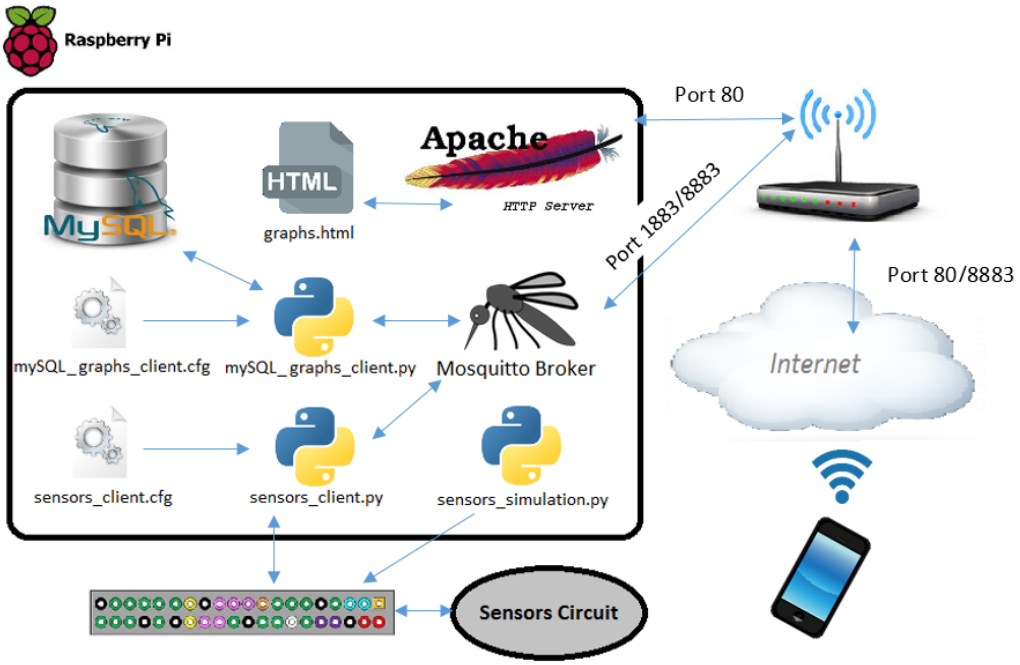
Λαμβάνοντας υπόψη τη θέληση των πλείστων ανθρώπων για διευκόλυνση τις καθημερινότητας τους με χρήση της τεχνολογίας, και το γεγονός ότι τα ολοκληρωμένα συστήματα αυτοματισμού οικίας που υπάρχουν στην αγορά δεν είναι οικονομικά προσιτά σε όλους, δημιουργήθηκε αυτή η πλατφόρμα με υλικά χαμηλού κόστους και λογισμικό ανοιχτού κώδικα.

5.1 Αρχιτεκτονική Πλατφόρμας

Με πολύ απλά λόγια, η πλατφόρμα λειτουργεί ως εξής · οι αισθητήρες είναι τοποθετημένοι σε μια οικία και η πληροφορία για την τιμή-κατάσταση τους μαζεύεται σε μια κεντρική υπολογιστική μονάδα, το Raspberry Pi. Το Raspberry Pi συνδεδεμένο στο δρομολογητή (router) της οικίας επιτρέπει τον απομακρυσμένο έλεγχο του ιδίου, της πληροφορίας που παρέχουν οι αισθητήρες και συνεπώς της κατάστασης όλης της οικίας. Στα πλαίσια της εργασίας έχει δοθεί μεγάλη βαρύτητα στην υλοποίηση μιας εφαρμογής android για κινητά η οποία λειτουργεί ως διεπαφή για να έχει απομακρυσμένο έλεγχο ο χρήστης. Η εφαρμογή android θα αναλυθεί με λεπτομέρεια στο επόμενο κεφάλαιο. Επίσης, είναι σημαντικό να αναφερθεί ότι κάθε επί μέρους διαδικασία έχει υλοποιηθεί ξεχωριστά, με στόχο την εύκολη επέμβαση στον κώδικα, την καλύτερη κλιμάκωση, αλλά και τη δυνατότητα επαναχρησιμοποίησης σε άλλες εφαρμογές.

Αρχικά, κύριο κομμάτι της πλατφόρμας είναι ο υπολογιστής Raspberry Pi 2. Η δυνατότητα του για λειτουργία 24/7 και η υπολογιστική του ισχύ τον καθιστούν εξαιρετική επιλογή για να τρέχει το λογισμικό που χρειάζεται η πλατφόρμα. Το Raspberry Pi λειτουργεί κυρίως σαν Gateway. Συνοπτικά είναι εγκατεστημένα τα εξής λογισμικά:

- Ο **Mosquitto Broker**, πυρήνας όλης της επικοινωνίας στην πλατφόρμα.
- Η βάση δεδομένων **MySQL**, στην οποία αποθηκεύονται όλα τα ιστορικά δεδομένα για τους αισθητήρες.
- Ο **Apache Server**, στον οποίο τρέχει μια ιστοσελίδα για παρουσίαση ιστορικών στοιχείων σε μορφή γραφικών παραστάσεων.



Σχήμα 5.1: Αρχιτεκτονική Πλατφόρμας

Στην συνέχεια, για να λειτουργήσει το όλο σύστημα υπάρχουν 3 MQTT clients. Υπάρχει ο client ο οποίος διαβάζει την κατάσταση των αισθητήρων οι οποίοι έχουν συνδεθεί στα GPIO pins του Raspberry Pi και κάνει publish την πληροφορία στο broker. Επίσης ο ίδιος κάνει subscribe σε topics που αφορούν output συσκευές (π.χ διακόπτες) ώστε να προβεί στις κατάλληλες ενέργειες όταν λάβει εντολή. Ο client αυτός τρέχει στο sensors_client.py.

Ο επόμενος client έχει ως κύρια λειτουργία την αποθήκευση οποιασδήποτε πληροφορίας περνά από το broker και αφορά την κατάσταση στις συσκευές. Επιπλέον είναι υπεύθυνος για τη δημιουργία γραφικών παραστάσεων που αιτείται ο χρήστης μέσω διεπαφής σε συγκεκριμένο topic. Ο client αυτός τρέχει στο mySQL_graphs_client.py.

Ο τρίτος client στην ουσία δεν είναι μοναδικός. Είναι μια ομάδα από clients οι οποίοι δημιουργούνται από το/τους χρήστη/χρήστες μέσω της εφαρμογής android για παρακολούθηση και έλεγχο της οικίας. Η πληροφορία που δείχνουν οι clients αυτοί, προσαρμόζεται στις επιλογές του χρήστη. Περισσότερα στο επόμενο κεφάλαιο, στο οποίο αναλύεται η εφαρμογή android.

Τόσο το εκτελέσιμο sensors_client.py όσο και το mySQL_graphs_client-.py έχουν αρχεία ρυθμίσεων, το sensors_client.cfg και το mySQL_graphs_-client.cfg αντίστοιχα. Τα αρχεία αυτά λειτουργούν ως διεπαφή για το χρήστη, στα οποία μπορεί να ρυθμίσει τους δυο clients και τις λειτουργίες τους. Επίσης βοηθάνε στην κλιμάκωση της εφαρμογής και αποφεύγουν την αναγκαιότητα να έχει προγραμματιστικές ικανότητες ο χρήστης.

Ο αναγκαίος κώδικα για να παρουσιάζονται οι γραφικές παραστάσεις σε ιστοσελίδα υλοποιήθηκε στο αρχείο graphs.html.

Στην αρχή της εργασίας ήταν αναγκαία η ύπαρξη μετρήσεων ώστε να ελεγχθεί η ροή δεδομένων σε αυτή καθώς και η ευρύτερη λειτουργία της. Για το σκοπό αυτό, υλοποιήθηκε ένας στοιχειώδης προσομοιωτής αισθητήρων, το εκτελέσιμο sensors_simulation.py. Ακολούθως, προστέθηκαν και μερικοί πραγματικοί αισθητήρες για ολοκλήρωση της πλατφόρμας.

Όσα αναφέρθηκαν πιο πάνω μπορεί να τα παρατηρήσει κανείς στο σχεδιάγραμμα 5.1. Στην συνέχεια θα γίνει πιο λεπτομερής ανάλυση κάθε διαδικασίας και προβλημάτων που αντιμετωπίστηκαν κατά την υλοποίηση της πλατφόρμας.

5.2 Προώθηση θύρας (Port Forwarding)

Στα δίκτυα υπολογιστών, η προώθηση ή η χαρτογράφηση θύρας είναι μια εφαρμογή μετάφρασης διευθύνσεων δικτύου (NAT) η οποία ανακατευθύνει ένα αίτημα επικοινωνίας από ένα συνδυασμό διεύθυνσης και θύρας σε ένα άλλο, όταν τα πακέτα διέρχονται από μια πύλη δικτύου (Gateway).[18]

Συγκεκριμένα στην παρούσα εφαρμογή χρειάστηκε για να λυθεί το εξής πρόβλημα. Ο δρομολογητής σε κάθε οικία έχει μια μοναδική διεύθυνση (IP address), ώστε να μπορεί να στέλνει και να λαμβάνει δεδομένα από το διαδίκτυο. Επιπλέον, οι συσκευές που συνδέονται στο διαδίκτυο μέσω αυτού του δρομολογητή, δημιουργούν ένα εσωτερικό δίκτυο και στη κάθε μια προσδίδεται μια μοναδική (στο εσωτερικό δίκτυο) διεύθυνση. Οι διευθύνσεις αυτές συνήθως είναι της μορφής 192.168.1.X (X=[1,255]). Έστω ότι, σε μια συσκευή στο εσωτερικό δίκτυο είναι εγκατεστημένος ένας web server και τρέχει μια ιστοσελίδα. Πως μπορεί να έχει απομακρυσμένη πρόσβαση στην ιστοσελίδα κάποιος χρήστης; Ο χρήστης αυτό που χρειάζεται να ξέρει είναι την διεύθυνση του δρομολογητή και σε ποια θύρα τρέχει η ιστοσελίδα, ώστε να στείλει το κατάλληλο αίτημα. Πώς όμως ο δρομολογητής ξέρει σε ποία από τις εσωτερικές συσκευές θα προωθήσει το αίτημα; Εδώ είναι που χρειάζεται να γίνει προώθηση θύρας σε συγκεκριμένη συσκευή. Για να γίνει προώθηση θύρας σε κάποιο δρομολογητή συνήθως υπάρχει κάποια επιλογή στις ρυθμίσεις του. Άρα είναι σαφές ότι θα διαφέρει από μοντέλο σε μοντέλο.

Σε αυτή την εργασία γίνονται δυο αιτήματα από το εξωτερικό δίκτυο στο εσωτερικό. Το πρώτο είναι οι MQTT clients οι οποίοι θέλουν να συνδεθούν στο Mosquitto Broker και το δεύτερο είναι η πρόσβαση στη σελίδα με τις γραφικές παραστάσεις. Έγινε χρήση των προεπιλεγμένων θυρών για κάθε λειτουργία. Για εξωτερική σύνδεση στο broker επιτρέπεται μόνο TLS/SSL σύνδεση στην προκαθορισμένη θύρα, 8883. Στο web server η προκαθορισμένη θύρα είναι η 80. Έγινε λοιπόν ρύθμιση στο δρομολογητή της οικίας ώστε τα αιτήματα σε αυτές τις θύρες να προωθούνται στο Raspberry Pi. Στην εικόνα 5.2 παρουσιάζονται οι ρυθμίσεις στο δρομολογητή της οικία όπου έγινε η υλοποίηση της

Enable	Name	WAN Start Port	LAN Host Start Port	External IP Address	Modify	Delete
	Protocol	WAN End Port	LAN Host End Port	LAN Host IP Address		
✓	raspberry_	22	22	178.59.198.180		
	TCP AND U	22	22	192.168.1.5		
✓	mosquitto	8883	8883	178.59.198.180		
	TCP AND U	8883	8883	192.168.1.5		
✓	web_serve	80	80	178.59.198.180		
	TCP AND U	80	80	192.168.1.5		

Σχήμα 5.2: Χαρτογράφηση θυρών στον τοπικό δρομολογητή

πλατφόρμας. Η προώθηση στη θύρα 22 έχει σκοπό την απομακρυσμένη SSH σύνδεση στο Raspberry Pi.

5.3 Ρύθμιση Mosquitto Broker

Η εγκατάσταση του Mosquitto Broker στο Raspberry Pi είναι πολύ απλή και οι προκαθορισμένες ρυθμίσεις επιτρέπουν ήδη την εκτέλεση των εντολών publish και subscribe μέσω της θύρας 1883. Για αυτή τη πλατφόρμα, υλοποιήθηκαν πιο προχωρημένες ρυθμίσεις, οι οποίες αφορούν κυρίως την ασφάλεια.

Η πρώτη ρύθμιση που έγινε ήταν να τεθούν εξουσιοδοτημένοι χρήστες στο broker. Με την εντολή mosquitto_passwd δημιουργείται ένας νέος χρήστης. Για παράδειγμα εκτελώντας mosquitto_passwd -c /etc/mosquitto/-passwdfile myuser, θα ακολουθήσει επιλογή για εισαγωγή κωδικού και ο νέος χρήστης, myuser, θα καταχωρηθεί στο αρχείο passwdfile.

Αφού δημιουργηθεί το αρχείο με τους εξουσιοδοτημένους χρήστες πρέπει να ενημερωθεί ο broker, ότι θα δέχεται συνδέσεις μόνο από χρήστες που είναι καταχωρημένοι στο αρχείο. Αυτό γίνεται εισάγοντας στο αρχείο ρυθμίσεων του broker, mosquitto.conf, τις γραμμές password_file /etc/mosquitto/conf.d/-passwdfile και allow_anonymous false. Η πρώτη καθορίζει το αρχείο στο οποίο έχουν δηλωθεί οι εξουσιοδοτημένοι χρήστες και η δεύτερη επιβάλει στο broker να απορρίπτει τη σύνδεση σε ανώνυμους, μη εξουσιοδοτημένους χρήστες.

Ακολούθως, εισάγοντας τις ρυθμίσεις persistence true και persistence_-location /var/lib/mosquitto/, επιτρέπεται στους clients να κάνουν επίμονες συνδέσεις και καθορίζεται σε ποιο φάκελο ο Mosquitto Broker θα διατηρεί βάση δεδομένων. Στη βάση δεδομένων αποθηκεύονται τα στοιχεία (π.χ. client_id) και η πληροφορία (π.χ. subscriptions και last will message) που αφορά clients οι οποίοι έχουν δηλώσει τη μεταβλητή cleanSession ψευδή.

Στη συνέχεια έγινε ρύθμιση, ώστε clients μέσω του εξωτερικού δικτύου να μπορούν να συνδεθούν μόνο με TLS/SSL σύνδεση. Η TSL/SSL σύνδεση και οι λόγοι χρήσης της, αναλύθηκαν σε προηγούμενο κεφάλαιο. Με την ρύθμιση listener 8883, ο broker μπορεί να δέχεται και συνδέσεις στη θύρα 8883 και καθορίζοντας τα αρχεία cafile, certfile και keyfile, οι συνδέσεις στη θύρα 8883 θα πρέπει να παρουσιάσουν τα αντίστοιχα πιστοποιητικά, για να εγκριθεί η σύνδεση.

Για να φτιαχτούν τα αρχεία cafile, certfile και keyfile χρησιμοποιήθηκε το λογισμικό OpenSSL. Ξεκινώντας από το τελευταίο, το keyfile, είναι το αρχείο το οποίο περιέχει το private key του broker. Το certfile περιέχει βασικά στοιχεία για το broker (όνομα, οργανισμό, διεύθυνση κλπ), κάτι αντίστοιχο με ταυτότητα, και το public key. Για να δημιουργηθεί χρειάζεται το private key. Ακολούθως το certfile κατατίθεται σε μια αρχή (Certificate Authority), η οποία θα παράξει το τελικό SSL πιστοποιητικό (cafile). Το SSL πιστοποιητικό δεν περιέχει το private key. Όπως αναφέρθηκε το ρόλο αυτής της αρχής (CA) έπαιξε το λογισμικό OpenSSL.

Με τα πιο πάνω βεβαιώθηκε ότι κάθε σύνδεση στη θύρα 8883 θα έχει κρυπτογραφημένη επικοινωνία. Επεκτείνοντας την ασφάλεια, υπάρχει η επιλογή να υποχρεωθεί ο client κατά την SSL χειραψία, να παρουσιάσει το κατάλληλο πιστοποιητικό για τη ταυτότητά του. Κάτι αντίστοιχο με το username/password. Τέτοια ρύθμιση ενεργοποιείται προσθέτοντας τη γραμμή require_certificate true. Αυτό το πιστοποιητικό παράγεται και πάλι από το λογισμικό OpenSSL και χρειάζεται τα στοιχεία του broker, για να έχει νόημα η όλη διαδικασία. Αν επιλεγεί αυτός ο τρόπος πιστοποίησης, με την επιλογή use_identity_as_username true θα αγνοηθεί το password_file για αυτή τη θύρα. Να σημειωθεί ότι αυτός ο πλεονασμός προσφέρει περισσότερη ασφάλεια χωρίς ιδιαίτερη επιβάρυνση.

5.4 Αρχείο Ρυθμίσεων ως διεπαφή

Για καλύτερη εμπειρία χρήσης της πλατφόρμας για τα 2 εκτελέσιμα, sensors_client.py όσο και mySQL_graphs_client.py υπάρχουν αρχεία ρυθμίσεων.

Στο sensors_client.cfg ο χρήστης μπορεί να εισάγει τις ρυθμίσεις για την MQTT σύνδεση. Επίσης μπορεί να εισάγει νέες συσκευές και ανάλογα με το είδος της, ενημερώνει το εκτελέσιμο σε ποιο pin του Raspberry Pi την έχει συνδέσει ή σε ποιο αρχείο θα βρει τις μετρήσεις της. Στην εικόνα 5.3 παρουσιάζεται πως έχει συμπληρωθεί το αρχείο στις δοκιμές της πλατφόρμας.

Στο άλλο αρχείο ρυθμίσεων, το mySQL_graphs_client.cfg, ο χρήστης μπορεί να εισάγει τις ρυθμίσεις για την MQTT σύνδεση καθώς και τη σύνδεση στην βάση δεδομένων MySQL. Επιπλέον δηλώνονται οι λίστες με τις συσκευές που είναι συνδεδεμένες στην πλατφόρμα, ανάλογα με το είδος τους. Στην εικόνα 5.4 παρουσιάζεται πως έχει συμπληρωθεί το αρχείο στις δοκιμές της πλατφόρμας.

```
[Mqtt Client Details]
#Settings for mqtt connection
ClientId="SensorsClient"
Server="localhost"
Port=1883
Clean Session=True
#Optional
Time Out=5
Keep Alive Timeout=200
UserName="myname"
Password="mypass"
SSL=False
CA Cert Path=

[Raspberry Pi 2 - Input Sensors Mapping]

    [[Two State - Input list]]
    #List of input sensors (two state) in the following form: topic:GPIO pin number
    Home/Hall/Window_1=18
    Home/Hall/Window_2=25

    [[Multi State - Input list]]
    #List of input sensors (multi state) in the following form: topic: input file
    Home/Hall/Temperature="/sys/bus/w1/devices/28-000004a7b5bb/w1_slave"

[Raspberry Pi 2 - Output Sensors Mapping]

    [[Two State - Output list]]
    #List of output sensors (two state) in the following form: topic:GPIO pin number
    Home/Hall/Heater_Switch=24
    Home/LivingRoom/Light_Switch=23
```

Σχήμα 5.3: Το αρχείο ρυθμίσεων sensors_client.cfg

```
[Mqtt Client Details]

ClientId="DatabaseClient"
Server="localhost"
Port=1883
Clean Session=True
#Optional
Time Out=5
Keep Alive Timeout=200
UserName="myname"
Password="mypass"
SSL=False
CA Cert Path=

[MySql Details]

Server="localhost"
Port=
Username="root"
Password=""
Database="Sensors"

[Other]

TwoStateSensorList= "Home/Hall/Window_1,Home/Hall/Window_2,Home/Hall/Heater_Switch,
Home/LivingRoom/Light_Switch"
MultiStateSensorList="Home/Hall/Temperature"
```

Σχήμα 5.4: Το αρχείο ρυθμίσεων mySQL_graphs_client.cfg

5.5 Από τα GPIO pins στον Broker και αντίστροφα

Όπως έχει ήδη αναφερθεί, το εκτελέσιμο sensors_client.py είναι υπεύθυνο για τη κατάλληλη επεξεργασία και μεταφορά της πληροφορίας από τους αισθητήρες, που είναι συνδεδεμένοι στα GPIO pins του Raspberry Pi στο Broker και αντίστροφα.

Αναλυτικότερα είναι ένα αρχείο γραμμένο στην γλώσσα προγραμματισμού python και οι 2 κύριες βιβλιοθήκες που χρησιμοποιεί είναι η paho-mqtt και RPi.GPIO. Η πρώτη είναι μια βιβλιοθήκη για δημιουργία και εκτέλεση των βασικών λειτουργιών ενός MQTT client, η οποία παρέχεται από το έργο Eclipse Paho. Η δεύτερη περιέχει τις απαραίτητες συναρτήσεις για έλεγχο των GPIO pins του Raspberry Pi.

Αυτό το εκτελέσιμο θα λάβει της ρυθμίσεις του από το αρχείο ρυθμίσεων sensors_client.cfg. Θα προσπαθήσει να συνδεθεί στο Mosquitto Broker και σε περίπτωση αποτυχίας, θα τερματίσει την εκτέλεση του με κατάλληλο μήνυμα. Σε περίπτωση επιτυχίας, θα εκτελέσει subscribe σε όλα τα topics, για τα οποία έχουν οριστεί αισθητήρες εξόδου. Αισθητήρες δηλαδή για τους οποίους η τιμή θα αλλάζει κατόπιν εντολής του χρήστη στο ανάλογο topic. Ακολούθως, θα ρυθμίσει τις πιο κάτω συναρτήσεις, οι οποίες λειτουργούν ασύγχρονα και παράλληλα μεταξύ τους:

- on_message - Είναι η συνάρτηση, η οποία θα λάβει τα μηνύματα από το broker. Τα μηνύματα είναι εντολές της μορφής "ON" ή "OFF". Το κάθε μήνυμα, αφορά κάποιο topic το οποίο με τη σειρά του αναπαριστά κάποια συσκευή, η οποία είναι γνωστό σε ποιο pin είναι συνδεδεμένη. Τελικά ανάλογα με το μήνυμα θα αλλάξει και η κατάσταση της αντίστοιχης συσκευής.

- pins_callback - Αυτή η συνάρτηση παρακολουθεί τα pins, τα οποία έχουν ορισθεί ως συσκευές εισόδου δυο καταστάσεων (ON/OFF). Κάθε φορά που παρατηρεί κάποια αλλαγή εκτελεί publish το κατάλληλο μήνυμα στο αντίστοιχο topic.

- readMultiStateSensors - Αυτή η συνάρτηση διαβάζει συνεχώς την τιμή των αισθητήρων που αποθηκεύουν τις μετρήσεις τους σε αρχεία. Τέτοιοι αισθητήρες είναι, οι αισθητήρες θερμοκρασίας, υγρασίας, έντασης φωτός κλπ. Όταν διαπιστωθεί σημαντική αλλαγή (π.χ μεταβολή της θερμοκρασία κατά ένα βαθμό Κελσίου) εκτελεί publish το κατάλληλο μήνυμα στο αντίστοιχο topic.

Η σύμβαση που ακολουθείται στα μηνύματα για τους αισθητήρες εισόδου είναι "κατάσταση συσκευής"@"χρονική στιγμή", όπου η χρονική στιγμή είναι η στιγμή που καταμετρήθηκε το γεγονός. Για παράδειγμα, για μια μαγνητική επαφή, όταν ενεργοποιηθεί, το μήνυμα θα είναι ON@2016-06-16 12:30:00.

5.6 Από τον Broker στη βάση δεδομένων

Είναι πολύ σημαντικό σε μια πλατφόρμα αυτοματισμού οικίας οι χρήστες να έχουν πρόσβαση σε ιστορικά στοιχεία της κατάστασης του σπιτιού. Για να μπορεί να συμβεί αυτό, πρέπει πρώτα αυτές οι αλλαγές να είναι αποθηκευμένες σε μια βάση δεδομένων.

Το εκτελέσιμο mySQL_grpahs_client.py είναι ένα αρχείο γραμμένο στην γλώσσα προγραμματισμού python. Αυτό το εκτελέσιμο θα λάβει τις ρυθμίσεις του από το αρχείο ρυθμίσεων mySQL_graphs_client.cfg. Θα προσπαθήσει να συνδεθεί στο Mosquitto Broker και στη βάση δεδομένων MySQL. Σε περίπτωση αποτυχίας θα τερματίσει την εκτέλεσή του με κατάλληλο μήνυμα. Η βασική του λειτουργία είναι να "παρακολουθεί" ό,τι πληροφορία περνά από το Mosquitto Broker και όταν αφορά αλλαγές στις συσκευές τις οικίας την αποθηκεύει στη βάση δεδομένων MySQL. Πιο συγκεκριμένα, για κάθε καινούργια συσκευή δημιουργείται ένας νέος πίνακας στη βάση δεδομένων. Αυτός ο πίνακας έχει όνομα το topic στο οποίο ανήκει η συσκευή και 3 πεδία: το μοναδικό αναγνωριστικό της κάθε εγγραφής, την κατάσταση της συσκευής και την χρονική στιγμή η οποία έγινε η καταμέτρηση. Ενδεικτικά στην εικόνα 5.5 είναι ένα στιγμιότυπο από μετρήσεις της συσκευής Home/Hall/Temperature.

Id	Time	State
357	2016-06-16 02:33:15	29
358	2016-06-16 03:34:08	28
359	2016-06-16 04:35:00	28
360	2016-06-16 05:35:52	28
361	2016-06-16 06:36:44	29
362	2016-06-16 07:37:36	28
363	2016-06-16 08:38:28	28
364	2016-06-16 09:39:21	29
365	2016-06-16 10:40:13	29
366	2016-06-16 11:41:05	29
367	2016-06-16 12:41:57	29
368	2016-06-16 13:42:49	29
369	2016-06-16 14:43:42	29
370	2016-06-16 15:44:34	29
371	2016-06-16 16:45:26	29
372	2016-06-16 17:46:18	29
373	2016-06-16 18:47:10	30
374	2016-06-16 19:48:02	30
375	2016-06-16 20:48:55	29
376	2016-06-16 21:49:47	30
377	2016-06-16 22:50:39	30

Σχήμα 5.5: Στιγμιότυπο από μετρήσεις της συσκευής Home/Hall/Temperature.

5.7 Από τη βάση δεδομένων σε online Διαγράμματα

Το εκτελέσιμο mySQL_grpahs_client.py κάνει ακόμη μια σημαντική δουλειά. Δημιουργεί τις γραφικές παραστάσεις που αιτείται να δει ο χρήστης μέσω

της εφαρμογής android. Θεωρήθηκε μη αποδοτικό κάθε χρήστης που χρησιμοποιεί την εφαρμογή android, να έχει όλα τα ιστορικά στοιχεία στην μνήμη του κινητού ή να του αποστέλλονται σε οποιαδήποτε μορφή. Έτσι, επικράτησε η πιο κάτω διαδικασία.

Σε πρώτη φάση υπάρχει ως εσωτερικός μηχανισμός για τη πλατφόρμα το topic Home/Commands/Graph, στο οποίο οι MQTT clients στο mySQL_graphs_client.py και στην εφαρμογή κινητού ανταλλάζουν (μέσω του broker πάντα) εντολές. Ουσιαστικά και οι 2 clients εκτελούν subscribe και publish σε αυτό το topic. Για να γίνει πιο κατανοητό, ο client στη φορητή συσκευή στέλνει σε ένα μήνυμα το όνομα της συσκευής και τη χρονική περίοδο που ενδιαφέρεται. Το μήνυμα είναι της μορφής "Home/LivingRoom/Temperature,2016-06-12,2016-06-19", υποδηλώνοντας ότι θέλει να δει τη θερμοκρασία στο σαλόνι για εκείνη τη εβδομάδα. Με τη σειρά του ο client στο Raspberry Pi, διαβάζοντας αυτό το μήνυμα θα ζητήσει τα κατάλληλα δεδομένα από τη βάση, θα φτιάξει το διάγραμμα και θα το αποθηκεύσει με το όνομα graph.png. Στην συνέχεια θα στείλει, στο ίδιο πάντα topic, το μήνυμα "READY". Με την παραλαβή αυτού του μηνύματος η εφαρμογή android γνωρίζει ότι το διάγραμμα είναι έτοιμο και φορτώνει τη σελίδα graphs.html από το web server μέσα στην εφαρμογή.

Η σελίδα graphs.html είναι σχεδιασμένη να ανταποκρίνεται στις ανάγκες κάθε συσκευής (responsive web page). Χρησιμοποιούνται οι γλώσσες προγραμματισμού για το διαδίκτυο html και php. Η μόνη λειτουργία της είναι να φορτώνει το αρχείο-εικόνα graph.png το οποίο επανεγγράφεται σε κάθε νέα επιλογή του χρήστη.

5.8 Προσομοίωση αισθητήρων

Για τον καλύτερο έλεγχο της λειτουργίας της πλατφόρμας αυτοματισμού οικίας που υλοποιήθηκε, κρίθηκε αναγκαία η υλοποίηση κάποιου είδους προσομοίωσης αισθητήρων. Με αυτό το τρόπο μπόρεσε να γίνει καλύτερος έλεγχος της ροής δεδομένων και να ελεγχθεί η απόκριση της πλατφόρμας σε μεγάλο όγκο δεδομένων και σε αλλαγές με μικρό χρονικό διάστημα. Επιπλέον, έτσι αποφεύχθηκαν επιπλοκές που θα παρουσιάζονταν από το υλικό και θα χρειάζονταν πολύτιμο χρόνο να εντοπιστούν. Παρόλα αυτά, η προσομοίωση δεν έπρεπε να επηρεάζει τον τρόπο λειτουργίας των υπόλοιπων μηχανισμών στη πλατφόρμα.

Λαμβάνοντας τα πιο πάνω υπόψη υλοποιήθηκε το εκτελέσιμο sensors_simulation.py, γραμμένο και αυτό στη γλώσσα προγραμματισμού Python. Η λειτουργία του είναι σχετικά πολύ απλή. Διαβάζει από το αρχείο ρυθμίσεων sensors_client.cfg σε ποια pins του Raspberry Pi έχουν δηλωθεί ως αισθητήρες εισόδου δύο καταστάσεων (π.χ μαγνητικές επαφές), τα δηλώνει ως pins εξόδου και έτσι αλλάζει την κατάσταση τους τυχαία σε τυχαία χρονικά διαστήματα. Με αυτό το τρόπο το ηλεκτρικό σήμα στο Raspberry Pi αλλάζει πραγματικά και έτσι η πλατφόρμα δεν αντιλαμβάνεται την επέμβαση αυτού του εκτελέσιμου.

5.9 Πραγματικοί αισθητήρες

Με την ολοκλήρωση του λογισμικού που χρησιμοποιείται στη πλατφόρμα, προστέθηκαν και κάποια πραγματικά εξαρτήματα για τελικό έλεγχο και παρουσίαση του ολοκληρωμένου συστήματος.

Για αισθητήρες εισόδου πολλαπλών τιμών χρησιμοποιήθηκαν δύο αισθητήρες θερμοκρασίας DS18B20. Ο ένας τοποθετήθηκε στο χωλ και ένας κοντά σε ένα παράθυρο, ώστε θεωρητικά να τροφοδοτεί το σύστημα με την εξωτερική θερμοκρασία. Συνεπώς, έγιναν οι απαραίτητες ρυθμίσεις, ώστε να παρακολουθείται οι θερμοκρασία στα topics Home/Hall/Temperature και Home/OutDoor/Tempe-rature.

Επιπρόσθετα, τοποθετήθηκαν 2 Leds ως συσκευές εξόδου, με τη μια να αναπαριστά το φωτισμό στο σαλόνι (Home/LivingRoom/Light_Switch) και η άλλη, τον έλεγχο του θερμοσίφωνα (Home/Hall/Heater_Switch). Η επιλογή αυτή έγινε, γιατί δεν ήταν δυνατή η επέμβαση στο ηλεκτρικό δίκτυο της οικίας όπου υλοποιήθηκε η πλατφόρμα.

Τέλος, για συσκευή εισόδου δύο καταστάσεων, τοποθετήθηκε μια μαγνητική επαφή στην κύρια είσοδο της οικίας.

Κεφάλαιο 6

Εφαρμογή Android (Hm-Monitoring)

6.1 Περιγραφή

Η εφαρμογή android, με όνομα "Hm-Monitoring", αποτελεί το σημαντικότερο κομμάτι της εργασίας. Η "Hm-Monitoring" αποτελεί τη διεπαφή για τον έλεγχο και την παρακολούθηση της πλατφόρμας αυτοματισμού αυτής της εργασίας, αλλά υλοποιήθηκε με τρόπο ώστε να μην εξαρτάται από αυτήν. Ακολουθώντας απλές συμβάσεις, που αφορούν το MQTT, μπορεί κανείς να τη χρησιμοποιήσει και σε δικό του σύστημα αυτοματισμού οικίας, το οποίο έχει το MQTT ως πρωτόκολλο επικοινωνίας. Για την υλοποίηση της αφιερώθηκε περίπου το 65% του συνολικού χρόνου δουλειάς. Στόχος ήταν μια εφαρμογή, λειτουργική, φιλική προς το χρήστη και με χαμηλή κατανάλωση πόρων (μπαταρίας, μνήμης).

Συνοπτικά, στη "Hm-Monitoring" ο χρήστης έχει την επιλογή να δημιουργήσει ένα ή περισσότερους MQTT clients, σε broker της επιλογής του. Υποστηρίζει δυνατότητα σύνδεσης με πιστοποίηση χρήστη και SSL συνδέσεις. Ο χρήστης για κάθε client που δημιουργεί, ρυθμίζει για ποιες συσκευές θέλει να λαμβάνει ειδοποιήσεις από το γραφικό περιβάλλον της εφαρμογής. Δίνοντας έτσι, σε κάποιο βαθμό, προσαρμοστικότητα της εφαρμογής στα ενδιαφέροντα του χρήστη. Προφανώς, για κάθε επιλογή, πρέπει να υπάρχει η αντίστοιχη υποδομή στην οικία για να λαμβάνει μετρήσεις η εφαρμογή. Επιπλέον, η σύνδεση και η παρακολούθηση στα επιθυμητά topics υλοποιήθηκε ώστε να διατηρείται και να λειτουργεί, ανεξάρτητα αν η εφαρμογή βρίσκεται στο background ή στο foreground της φορητής συσκευής. Αυτό είναι εφικτό με τη χρήση Android Service. Επιπρόσθετα, έχει υλοποιηθεί σύστημα ειδοποιήσεων χρήστη (notifications), όταν υπάρχει αλλαγή κατάστασης μιας συσκευής. Άλλο ένα χαρακτηριστικό, είναι η δυνατότητα φόρτωσης γραφικών παραστάσεων, με ιστορικά στοιχεία των συσκευών. Τέλος, υπάρχει η επιλογή παρακολούθησης της ίδιας της εφαρμογής για το χρόνο που είναι ζωντανή, στην μορφή Log.

Για το λογότυπο της εφαρμογής χρησιμοποιήθηκε το λογισμικό Iconion 2.7 και είναι στην εικόνα 6.1.

Σχήμα 6.1: Λογότυπο της Εφαρμογής Android, Hm-Monitoring

6.2 Προδιαγραφές συσκευής και Permissions

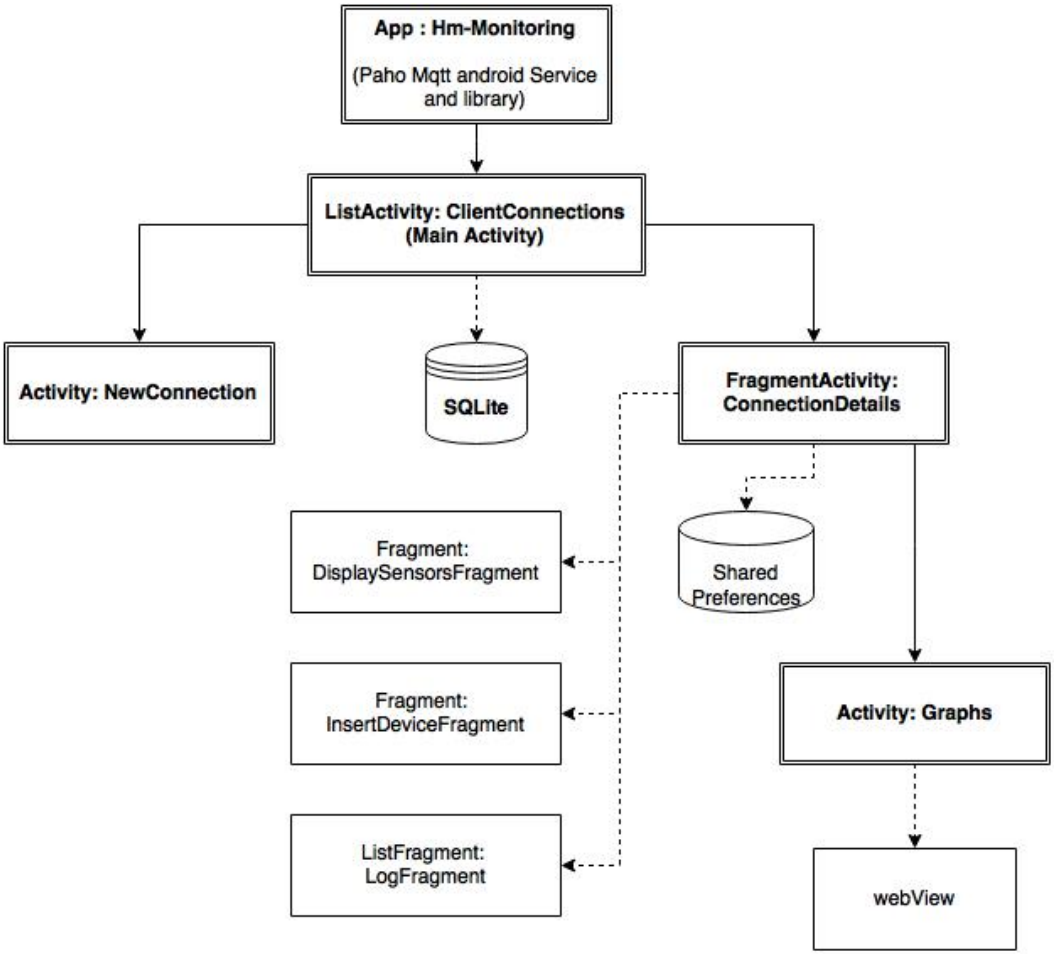
Η εφαρμογή για να λειτουργήσει πρέπει το API level του Android λογισμικού στη συσκευή να είναι τουλάχιστο 14 (android:minSdkVersion="14"). Επίσης χρειάζεται τα πιο κάτω permissions από τη συσκευή στην οποία θα τρέξει:

- WAKE_LOCK - Επιτρέπει στην εφαρμογή να χειρίζεται θέματα που αφορούν την κατανάλωση ενέργειας της συσκευής
- INTERNET - Επιτρέπει στην εφαρμογή να έχει πρόσβαση στο διαδίκτυο μέσω της συσκευής.
- WRITE_EXTERNAL_STORAGE - Επιτρέπει στην εφαρμογή να επεξεργάζεται δεδομένα στο χώρο εξωτερικής αποθήκευσης της συσκευής.
- ACCESS_NETWORK_STATE - Επιτρέπει στην εφαρμογή να διαβάζει την κατάσταση δικτύων της συσκευής.

6.3 Δομή Εφαρμογής

Η εφαρμογή "Hm-Monitoring" χρησιμοποιεί τις βιβλιοθήκες Mqtt Java client και Android Service, οι οποίες παρέχονται από το έργο Paho. Από τις βιβλιοθήκες αυτές έγινε χρήση κλάσεων και διεπαφών για σύνδεση σε MQTT broker. Επίσης, χρησιμοποιήθηκαν για υλοποίηση ασύγχρονου εσωτερικού μηχανισμού ο οποίος πραγματοποιεί επανασύνδεση των clients. Σε περίπτωση απροσδόκητης απώλειας σύνδεσης, παραλαμβάνει τα μηνύματα από τα επιθυμητά topics και ειδοποιεί το χρήστη για τα πάνω. Με τον όρο ασύγχρονος, εννοείται ότι οι λειτουργίες εκτελούνται σε ξεχωριστές background διεργασίες, ανεξάρτητα από τι είδους χρήση της συσκευής, κάνει ο χρήστης. Επιπλέον, η δομή της εφαρμογής είναι βασισμένη σε μια εφαρμογή-παράδειγμα που χρησιμοποιεί τις πιο πάνω βιβλιοθήκες, την Paho Andoid Example.

Συγκεκριμένα για τη δομή, η "Hm-Monitoring" αποτελείται από τέσσερα Activities στα οποία μπορεί να περιηγηθεί ο χρήστης. Το ClientConnections είναι το κύριο Activity της εφαρμογής, το οποίο είναι επέκταση του ListActivity. Σε αυτό, υπάρχει μια λίστα με όλες τις συνδέσεις που έχει κάνει ο χρήστης. Επιπλέον, αυτό το Activity είναι συνδεδεμένο με μια βάση δεδομένων (στην android συσκευή), την SQLite, για μόνιμη αποθήκευση των συνδέσεων που



Σχήμα 6.2: Δομή εφαρμογής

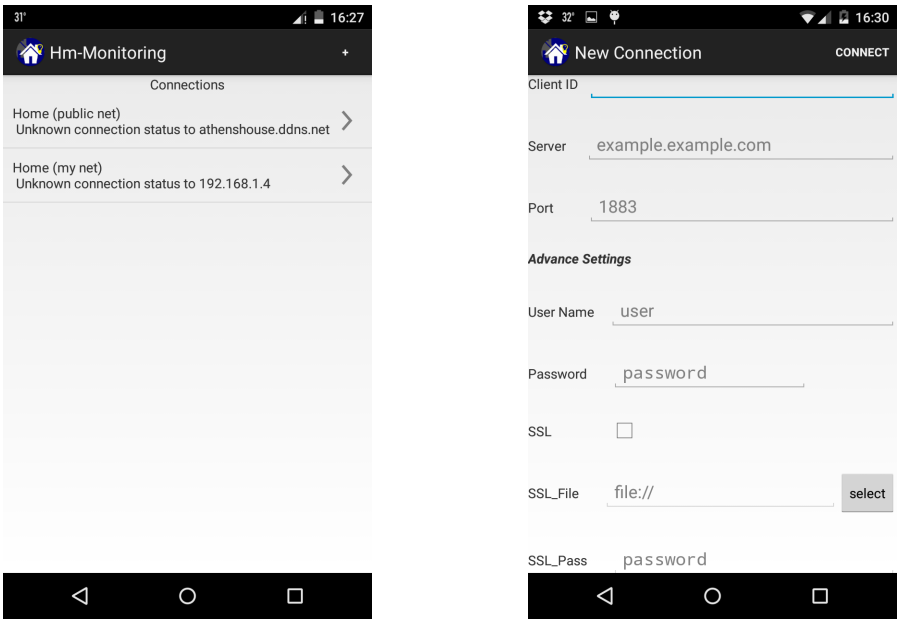
έχει προσθέσει ο χρήστης. Από το ClientConnections μπορεί να μεταβεί στο NewConnection ή στο ConnectionDetail. Στο 1^o, ο χρήστης συμπληρώνει μια φόρμα με τις ρυθμίσεις της νέας σύνδεσης που επιθυμεί και την προσθέτει στη λίστα του κύριου Activity. Το 2^o είναι μια επέκταση του FragmentActivity. Ο χρήστης μεταβαίνει σε αυτό, επιλέγοντας κάποια σύνδεση και είναι το περιβάλλον στο οποίο προσθέτει και ελέγχει τις συσκευές της πλατφόρμας, στην οποία ανήκει η σύνδεση. Στο ConnectionDetails υπάρχουν 3 fragments, το DisplaySensorsFragment, το InsertDeviceFragment και το LogFragment, το κάθε ένα με τις δικές του λειτουργίες. Οι ρυθμίσεις και οι επιλογές του χρήστη σε αυτό το Activity αποθηκεύονται μόνιμα με τη χρήση SharedPreferences. Από το ConnectionDetail, μπορεί κανείς να ανοίξει το Graphs Activity, στο οποίο μπορεί να ανατρέξει σε ιστορικά στοιχεία υπό τη μορφή γραφικών παραστάσεων. Για να πετύχει αυτό, χρησιμοποιείτε ένα webView στοιχείο.

Στην εικόνα 6.2 παρουσιάζονται σε σχεδιάγραμμα τα όσα περιγράφτηκαν πιο πάνω.

6.4 Λεπτομερής Παρουσίαση

Στην συνέχεια θα γίνει μια λεπτομερής παρουσίαση της εφαρμογής, με στιγμιότυπα από χρήση της πλατφόρμας σε πραγματικό χρόνο.

Ξεκινώντας την εφαρμογή, ο χρήστης βλέπει μια λίστα με προηγούμενες συνδέσεις, οι οποίες φορτώθηκαν από τη βάση δεδομένων. Για παράδειγμα βλέπει την εικόνα 6.3α΄, στην οποία ο χρήστης χρησιμοποιεί 2 συνδέσεις. Η μία είναι για σύνδεση στη πλατφόρμα από το εξωτερικό δίκτυο και η άλλη από το εσωτερικό.



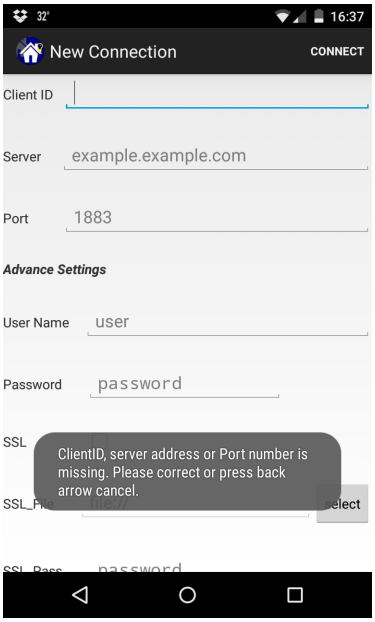
(α΄) Το κύριο Activity της εφαρμογής (β΄) Η φόρμα για πρόσθεση νέας σύνδεσης

Σχήμα 6.3: Main Activity και Φορμα νέας σύνδεσης

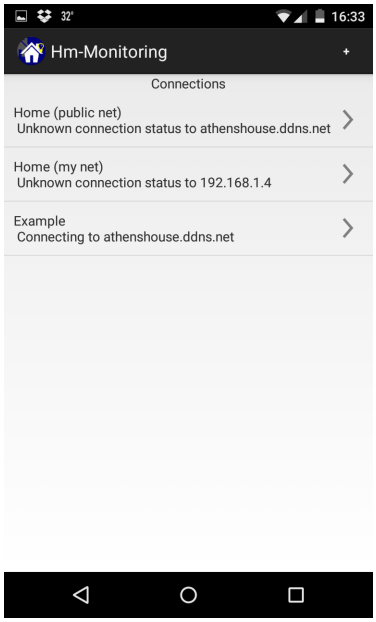
Για να προσθέσει μια νέα σύνδεση πατάει πάνω δεξιά, '+' σύμβολο, και μεταβαίνει στην εικόνα 6.3β΄. Εδώ, συμπληρώνει τα απαραίτητα στοιχεία για σύνδεση σε MQTT broker. Εμφανής είναι και η δυνατότητα για SSL σύνδεση, καθώς και η φόρτωση των απαραίτητων αρχείων για πιστοποίηση κατά την SSL χειραψία. Σημαντική παρατήρηση, τα πιστοποιητικά για να είναι δεκτά από το λογισμικό Android, πρέπει να μετατραπούν σε .bks μορφή και συνήθως συνοδεύονται από κωδικό. Αυτή η μετατροπή μπορεί να γίνει με το λογισμικό Keytool. Στην συνέχεια, επιλέγοντας CONNECT (πάνω δεξιά), θα αξιολογηθεί αν η φόρμα έχει συμπληρωθεί σωστά. Απαραίτητα είναι τα πεδία Client ID, Server και Port. Σε περίπτωση αποτυχίας, θα προβληθεί κατάλληλο μήνυμα (εικόνα 6.4α΄). Αντίθετα, θα επιτρέψει στο κύριο Activity, όπου η εφαρμογή θα προσπαθήσει να πραγματοποιήσει τη σύνδεση για πρώτη φορά (εικονα 6.4β΄).

Συνεχίζοντας, ο χρήστης θα πρέπει να αρχικοποιήσει το client με τις συσκευές που επιθυμεί να παρακολουθεί και να ελέγχει. Να σημειωθεί ότι, οποιαδήποτε εργασία στο client μπορεί να πραγματοποιηθεί μόνο όταν είναι συνδε-

δεμένος. Σε αντίθετη περίπτωση, όλες οι επιλογές είναι ανενεργές. Η σύνδεση και η αποσύνδεση μπορεί να γίνει από επιλογή που παρέχεται στο πάνω-δεξιά μέρος της οθόνης. Στις εικόνες 6.5 και 6.6 παρουσιάζεται η κατάσταση του ConnectionDetails Activity σε περίπτωση επιτυχημένης και αποτυχημένης σύνδεσης αντίστοιχα, πριν οποιαδήποτε ρύθμιση.

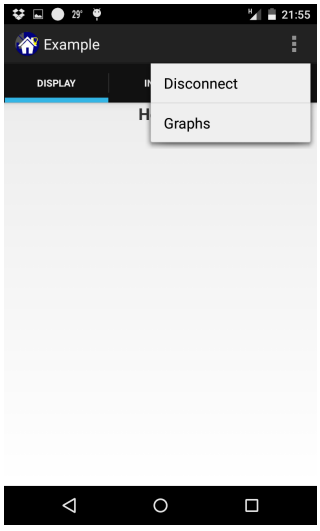


(α′) Μήνυμα σφάλματος στη φόρμα

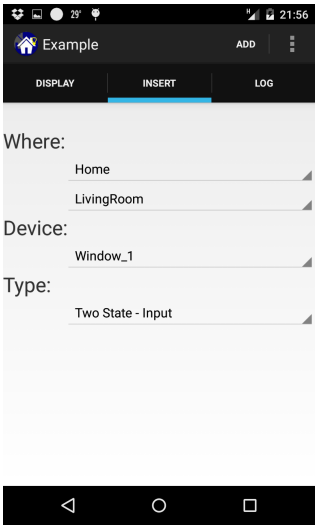


(β′) Σύνδεση νέου client

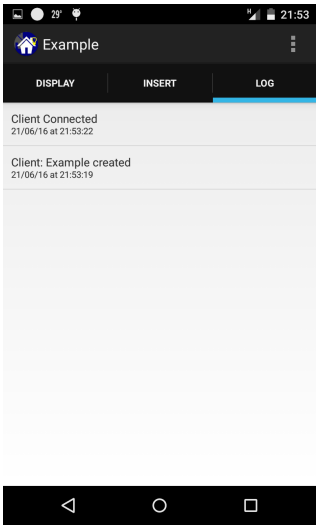
Σχήμα 6.4: Προσπάθεια προσθήκης client



(α′) Η καρτέλα Display



(β′) Η καρτέλα Insert



(γ′) Η καρτέλα Log

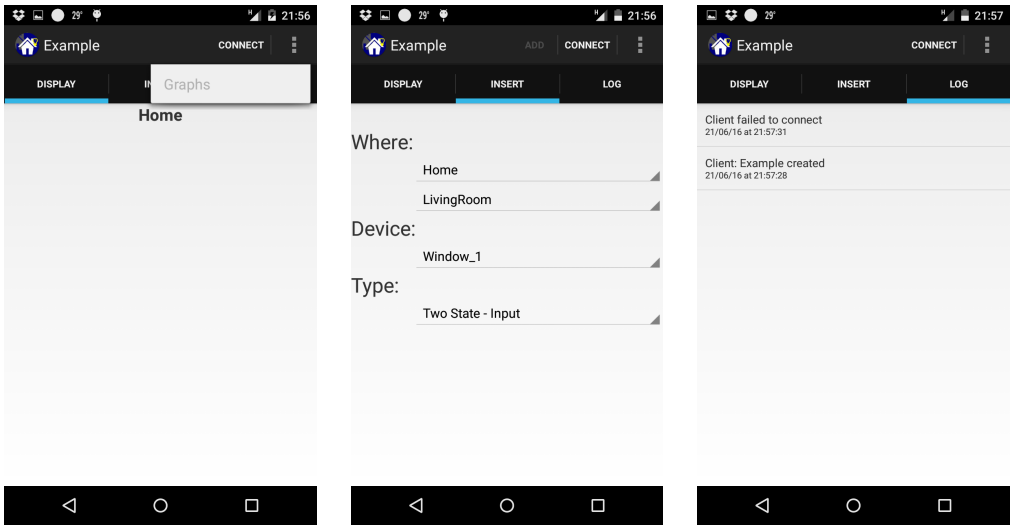
Σχήμα 6.5: Η ConnectionDetails Activity με επιτυχημένη σύνδεση

Παρατηρεί κανείς ότι στο ConnectionDetails Activity, τα fragments, DisplaySensorsFragment, InsertDeviceFragment και LogFragment παρουσιάζονται υπό μορφή καρτελών (tabs) και ο χρήστης μπορεί να περιηγηθεί σε αυτά είτε πατώντας στη επικεφαλίδα τους είτε σύροντας την οθόνη αφής δεξιά-αριστερά.

Οπότε, ο χρήστης για να προσθέσει συσκευές πρέπει να είναι συνδεδεμένος και να μεταβεί στη καρτέλα Insert. Εκεί, καλείται να επιλέξει το χώρο που βρίσκεται η συσκευή, τη συσκευή και τον τύπο της. Η εφαρμογή υποστηρίζει συσκευές εισόδου δύο καταστάσεων (ON/OFF) ή πολλαπλών καταστάσεων (π.χ αισθητήρα θερμοκρασίας). Επιπλέον υποστηρίζει την ενεργοποίηση ή απενεργοποίηση συσκευών, δηλαδή τον έλεγχο σε διακόπτες ON/OFF. Να σημειωθεί ότι η εφαρμογή δίνει την επιλογή σε μεγάλο εύρος συσκευών, αλλά επαναλαμβάνεται ότι η επιλογή τους προϋποθέτει την κατάλληλη υποδομή και ρύθμιση στην άλλη άκρη της πλατφόρμας. Για κάθε συσκευή που προσθέτει ο χρήστης, εκτελείται το κατάλληλο subscribe στο αντίστοιχο topic. Τα topics που δημιουργούνται είναι της μορφής Location/Room/Device, και είναι σύμβαση που αναμένεται να τηρείται σε όλη τη πλατφόρμα. Στην εικόνα 6.7 παρουσιάζονται οι λίστες επιλογών για το χρήστη.

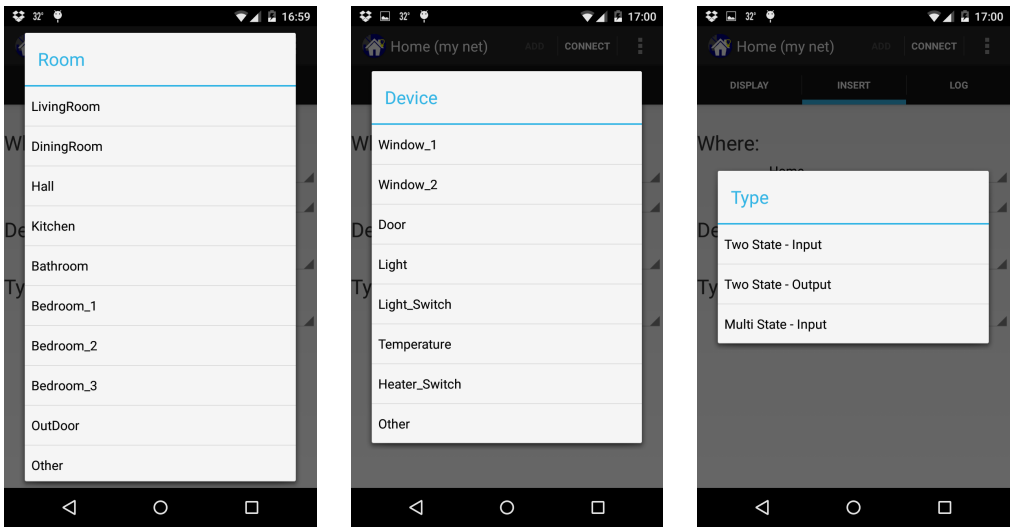
Ακολούθως, στην εικόνα 6.8α΄ υπάρχει ένα στιγμιότυπο της καρτέλας Display με πραγματικές μετρήσεις από τις δοκιμές της πλατφόρμας. Εκεί, μπορεί κανείς να δει την τελευταία, γνωστή στην εφαρμογή, τιμή της συσκευής καθώς και την ώρα που καταμετρήθηκε. Η εφαρμογή αναμένει να λαμβάνει μηνύματα της μορφής "κατάσταση συσκευής"@"χρονική στιγμή".

Καλείται ο αναγνώστης, να παρατηρήσει στις εικόνες 6.8α΄ και 6.8β΄ την απενεργοποίηση της δυνατότητας να αλλάξει την κατάσταση του διακόπτη, όταν ο client είναι αποσυνδεδεμένος. Επιπλέον, σημαντικό είναι να αναφερθεί και ο μηχανισμός που υλοποιήθηκε για αλλαγή κατάστασης διακόπτη. Ο διακόπτης είναι και αυτός συσκευή στη πλατφόρμα, με αντίστοιχο topic. Έτσι, όταν προστίθεται στις συσκευές, στην "Hm-Monitoring", εκτελείται subscribe στο topic της. Με αυτό το τρόπο μπορεί να ελεγχθεί αν όντως το μήνυμα του χρήστη έχει σταλεί στο broker. Το σημαντικότερο όμως, είναι ότι πολλοί χρήστες μπορούν να αλλάζουν τον ίδιο διακόπτη και να λαμβάνουν αυτές τις αλλαγές, ώστε να είναι ενήμεροι για την πραγματική τιμή του διακόπτη.



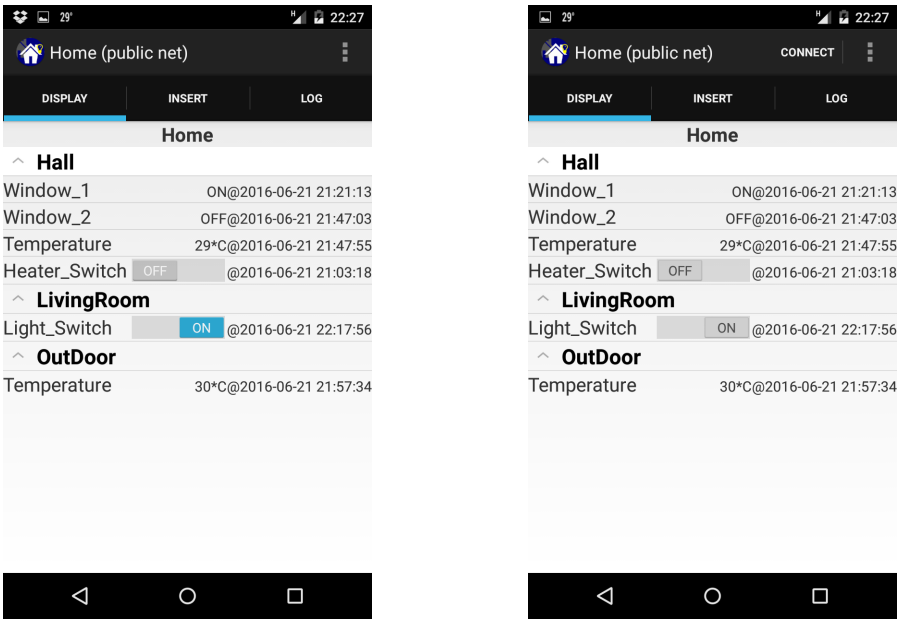
(α΄) Η καρτέλα Display (β΄) Η καρτέλα Insert (γ΄) Η καρτέλα Log

Σχήμα 6.6: Η ConnectionDetails Activity με αποτυχημένη σύνδεση



(α′) Επιλογή τοποθεσίας (β′) Επιλογή συσκευής (γ′) Είδος συσκευής

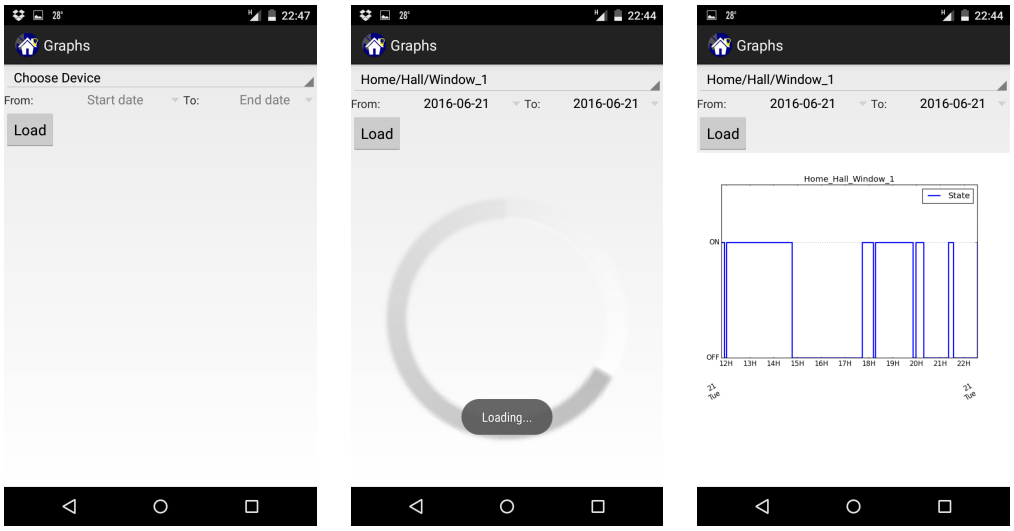
Σχήμα 6.7: Προσθήκη νέας συσκευής



(α′) Ενεργή σύνδεση (β′) Ανενεργή σύνδεση

Σχήμα 6.8: Στιγμιότυπο της καρτέλας Display με πραγματικές μετρήσεις

Ο χρήστης, από το ConnectionDetails Activity μπορεί να μεταβεί στο Graphs Activity, με την επιλογή που θα βρει πάνω-δεξιά. Προϋπόθεση και πάλι είναι η σύνδεση του client. Στο Graphs Activity μπορεί να επιλέξει τη συσκευή και τη χρονική περίοδο για την οποία επιθυμεί να δει ιστορικά στοιχεία. Τότε, σε ένα webView θα φορτωθεί η αντίστοιχη γραφική παράσταση. Παράδειγμα αυτής της διαδικασίας υπάρχει στην εικόνα 6.9.



(α′) Graph Activity (β′) Φόρτωση δεδομένων (γ′) Αποτέλεσμα

Σχήμα 6.9: Graph Activity - Αναζήτηση στα ιστορικά στοιχεία των συσκευών

Κεφάλαιο 7

Σύνοψη και Επεκτάσεις

7.1 Σύνοψη

Στη διπλωματική αυτή εργασία, μελετήθηκε και αναπτύχθηκε μια πλατφόρμα αυτοματισμού οικίας, στην οποία υπάρχει δυνατότητα για απομακρυσμένο έλεγχο και παρακολούθηση μέσω εφαρμογής Android. Αποτελούσε προτεραιότητα, η δημιουργία μιας εφαρμογής Android, η οποία να είναι αντάξια των προσδοκιών του χρήστη. Ο λόγος είναι ότι, η καλή λειτουργία της Hm-Monitoring, αποτελεί την κυριότερη μετρική για τη λειτουργικότητα όλης της πλατφόρμας.

Την χρονική περίοδο, από την ολοκλήρωση της όλης πλατφόρμας μέχρι τη συγγραφή αυτού του κειμένου, εξετάστηκε έμπρακτα η λειτουργία της, αφήνοντας τόσο το χρήστη όσο και τους ανθρώπους στους οποίους παρουσιάστηκε ικανοποιημένους. Η ευχρηστία και η φιλικότητα της εφαρμογής φορητών συσκευών, Hm-Monitoring, επιτεύχθηκαν.

7.2 Προτεινόμενες επεκτάσεις

Το σύνολο των λειτουργιών της πλατφόρμας σχεδιάστηκε και υλοποιήθηκε με όσο το δυνατό ανεξάρτητο τρόπο. Έτσι, δίνει την δυνατότητα να χρησιμοποιηθούν επιμέρους κομμάτια σε άλλες εφαρμογές. Επιπλέον, επιτρέπει την εύκολη επέκταση και κλιμάκωση της όλης εφαρμογής, τόσο σε μεγαλύτερες εγκαταστάσεις, όσο και στην πρόσθεση νέων ιδεών.

Μέσα από την εμπειρία υλοποίησης και χρήσης αυτής της εφαρμογής δημιουργήθηκαν κάποιες σκέψεις και προτάσεις για βελτίωση. Αρχικά από την πλευρά της κατασκευής στο Raspberry Pi, μπορούν να γίνουν 2 σημαντικές υλοποιήσεις, μια στο υλικό και μια στο λογισμικό.

Στην πλατφόρμα, μπορεί ήδη να προστεθούν μεμονωμένες ασύρματες ή ενσύρματες συσκευές. Με τη χρήση ηλεκτρονόμου, μπορεί να γίνει έλεγχος κάθε ηλεκτρονικής συσκευής. Αυτό όμως, δεν κλιμακώνει καλά ως προς το πλήθος των συσκευών. Ένα ξεχωριστό ασύρματο δίκτυο αισθητήρων, θα ήταν πιο εύκολο να τοποθετηθεί σε μια υπάρχουσα κατοικία και θα παρείχε στην πλατφόρμα, μεγάλη κλιμάκωση όσο αφορά των πλήθος των συσκευών που μπορεί να

συνδεθούν. Κάτι τέτοιο, μπορεί εύκολα να ρυθμιστεί και να προστεθεί στην ήδη υπάρχουσα υλοποίηση, με ένα παρόμοιο εκτελέσιμο με το sensors_client.py, στο οποίο θα αντιστοιχίζεται κάθε συσκευή με topic.

Ακολούθως, μπορεί να γίνει βελτίωση στα αρχεία ρυθμίσεων για τα εκτελέσιμα. Αντί αυτά, μπορεί να γίνει κάποια διεπαφή σε μορφή ιστοσελίδας. Σε αυτήν, ο χρήστης θα χρειάζεται εξουσιοδότηση, για να εισέλθει σε ένα γραφικό περιβάλλον, το οποίο θα είναι πιο εύχρηστο από την υπάρχουσα διεπαφή.

Τέλος, περιθώριο βελτίωσης υπάρχει και στον τρόπο παρουσίασης ιστορικών στοιχείων. Η υπάρχουσα ιστοσελίδα, graphs.html, μπορεί να αναπτυχθεί και να ανεξαρτητοποιηθεί εντελώς από την εφαρμογή Android. Να υλοποιηθεί μια ιστοσελίδα στην οποία, από τον υπολογιστή θα μπορούν εξουσιοδοτημένη χρήστες να ανατρέξουν σε ιστορικά των συσκευών, αλλά παράλληλα να είναι ανταποκρίσιμη (responsive) σε φορητές συσκευές, ώστε να εξακολουθήσει να τρέχει στο Graph Activity, της Hm-Monitoring.

Βιβλιογραφία

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, D. Boyle: From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence. Elsevier, 2014, ISBN 978-0-12-407684-6.

[2] Statista. Στατιστικά. [online]
http://www.statista.com/topics/2637/internet-of-things/

[3] Home Automation & Wiring (1 ed.). New York: McGraw-Hill/TAB Electronics. 1999-03-31. ISBN 9780070246744.

[4] Rye, Dave (October 1999). "My Life at X10". AV and Automation Industry eMagazine. AV and Automation Industry eMagazine. Retrieved October 8, 2014.

[5] Ericsson. Στατιστικά. [online]
https://www.ericsson.com/mobility-report

[6] Yahoo analytics. Στατιστικά. [online]
https://developer.yahoo.com/analytics/

[7] Cisco. Προβέψεις. [online].
http://www.cisco.com/c/dam/en_us/about/ac79/docs/Top_25_Predictions_121409rev.pdf

[8] Pål Kastnes. Ασφάλεια IoT. [online].
http://goo.gl/pCJj1D

[9] Oasis. MQTT Standard [online].
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[10] Github. MQTT libraries [online].
https://github.com/mqtt/mqtt.github.io/wiki/libraries

[11] Wikipedia. Raspberry Pi. [online].
https://en.wikipedia.org/wiki/Raspberry_Pi

[12] Raspberry Pi Foundation. What Is A Raspberry Pi? [Online].
https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[13] DS18B20. Datasheet. [online].
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[14] Apache. [online].
http://httpd.apache.org/ABOUT_APACHE.html

[15] Wikipedia. MySQL. [online].
https://el.wikipedia.org/wiki/MySQL

[16] Eclipse Paho. [online].
http://www.eclipse.org/paho/

[17] Wikipedia. Android. [online].
https://el.wikipedia.org/wiki/Android

[18] "Definition of: port forwarding". PC Magazine. Retrieved 2008-10-11.

[19] Github. Paho Android. [online].
https://github.com/eclipse/paho.mqtt.android

[20] HiveMq. Mqtt-Essentials. [online]. http://www.hivemq.com/blog/mqtt-essentials-wrap-up

Παράρτημα

A Raspberry Pi

Ακολουθούν τα αρχεία κώδικα που χρησιμοποιήθηκαν στο Raspberry Pi.

A.1 sensors_client.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from configobj import ConfigObj, ConfigObjError
import logging
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
import time

execfile("./sensors_helper.py")

loggingFilename="./sensors_client.log"
logging.basicConfig(filename=loggingFilename,format='%(asctime)s %(message)s',
    datefmt='%d/%m/%Y %I:%M:%S %p',level=logging.DEBUG)
logging.info("Started")

logging.info("Reading Configuration File...")
cfgFilename="./sensors_client.cfg"

try:
    config = ConfigObj(cfgFilename,unrepr=True, file_error=True)
except (ConfigObjError, IOError), e:
    logging.error( 'Could not read "%s": %s' % (cfgFilename, e))

clientId =  config['Mqtt Client Details']['ClientId']
server = config['Mqtt Client Details']['Server']
port = config['Mqtt Client Details']['Port']
cleanSession = config['Mqtt Client Details']['Clean Session']
timeOut = config['Mqtt Client Details']['Time Out']
keepAliveTimeout = config['Mqtt Client Details']['Keep Alive Timeout']
username = config['Mqtt Client Details']['UserName']
password = config['Mqtt Client Details']['Password']
SSL = config['Mqtt Client Details']['SSL']
ca_certs_path = config['Mqtt Client Details']['CA Cert Path']
twoStateInputMap = config['Raspberry Pi 2 - Input Sensors Mapping']['Two State -
    Input list']
multiStateInputMap = config['Raspberry Pi 2 - Input Sensors Mapping']['Multi State
     - Input list']
twoStateOutputMap = config['Raspberry Pi 2 - Output Sensors Mapping']['Two State -
     Output list']


twoStateInputPins= twoStateInputMap.values()
multiStateInputFiles= multiStateInputMap.values()
twoStateOutputPins = twoStateOutputMap.values()
twoStateInputRooms = twoStateInputMap.keys()
twoStateOutputRooms = twoStateOutputMap.keys()
multiStateInputRooms= multiStateInputMap.keys()

logging.info("Setting up Raspberry P...")

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(twoStateInputPins,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(twoStateOutputPins,GPIO.OUT)
for pin in twoStateInputPins:
    GPIO.add_event_detect(pin, GPIO.BOTH,callback=pins_callback,bouncetime=200)

logging.info("Setting up MQTT client...")
myData = None
client = mqtt.Client(clientId,cleanSession,myData,"MQTTv311")
client.on_connect = on_connect
client.on_message = on_message
client.on_publish = on_publish
```

```
client.max_inflight_messages_set(20)
client.message_retry_set(timeOut)

if SSL:
    client.tls_set(ca_certs_path,None,None,ssl.CERTR_REQUIRED,ss1.PROTOCOL_TLSv1,
        None)

if username!=None:
    client.username_pw_set(username,password)

client.user_data_set(myData)

logging.info("Connecting to broker...")

client.connect(server,port,keepAliveTimeout)


client.loop_start()
readMultiStateSensors()
#GPIO.remove_event_detect(twoStateInputPins)
GPIO.cleanup()
```

A.2 sensors_helper.py

```
def pins_callback(pin):
    # -*- coding: utf-8 -*-
    if GPIO.input(pin):
        status="ON@"+time.strftime('%Y-%m-%d %H:%M:%S')
    else:
        status="OFF@"+time.strftime('%Y-%m-%d %H:%M:%S')
    topic=twoStateInputRooms[twoStateInputPins.index(pin)]
    logging.info("Event detected: %s is %s",topic,status )
    (result,mid)=client.publish(topic,status,qos=2,retain=True)
    if result==0:
        logging.info("Message with id "+str(mid)+" sent")
    elif result == 1:
        logging.info("Message with id "+str(mid)+" not sent. CONNECTION LOST")

def on_connect(client, userdata, flags, rc):
    # The callback for when the client receives a CONNACK response from the server
        .
    logging.info("Connected with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    for i in twoStateOutputRooms:
        logging.info("Subscribing to topic: %s" % i)
        client.subscribe(i)

def on_message(client, userdata, msg):
    # The callback for when a PUBLISH message is received from the server.
    logging.info("Message received on "+msg.topic+" "+str(msg.payload))

    outputPin=twoStateOutputPins[twoStateOutputRooms.index(msg.topic)]
    value=msg.payload.split('@')[0]
    if value=="ON":
        GPIO.output(outputPin,1)
        logging.info(msg.topic+" set: "+ value)
    elif value=="OFF":
        GPIO.output(outputPin,0)
        logging.info(msg.topic+" set: "+ value)



def on_publish(client,userdata,mid):
    logging.info("Message with id "+str(mid)+" delivered")

def readMultiStateSensors():
    logging.info("reading multi state sensors")

    previous_temperature=0
    time_1 = time.time()
    while True:
```

```
        for thermFile in multiStateInputFiles:
            tfile = open(thermFile)
            text = tfile.read()
            tfile.close()
            temperature_data = text.split()[-1]
            temperature = float(temperature_data[2:])
            temperature = temperature / 1000
            time_2 = time.time()
            if (abs(temperature-previous_temperature)>1 or (time_2-time_1>3600)):
                time_1=time.time()
                status=str(int(round(temperature,0)))+"*C@"+time.strftime('%Y-%m-%
                    d %H:%M:%S')
                topic=multiStateInputRooms[multiStateInputFiles.index(thermFile)]
                logging.info("Event detected: %s is %s",topic,status )
                (result,mid)=client.publish(topic,status,qos=2,retain=True)
                if result==0:
                    logging.info("Message with id "+str(mid)+" sent")
                elif result == 1:
                    logging.info("Message with id "+str(mid)+" not sent.
                        CONNECTION LOST")
                previous_temperature=temperature
        time.sleep(60)
```

A.3 sensors_client.cfg

```
[Mqtt Client Details]
#Settings for mqtt connection
ClientId="SensorsClient"
Server="localhost"
Port=1883
Clean Session=True
#Optional
Time Out=5
Keep Alive Timeout=200
UserName="myname"
Password="mypass"
SSL=False
CA Cert Path=

[Raspberry Pi 2 - Input Sensors Mapping]

    [[Two State - Input list]]
    #List of input sensors (two state) in the following form: topic:GPIO pin
        number
    Home/Hall/Window_1=18
    Home/Hall/Window_2=25

    [[Multi State - Input list]]
    #List of input sensors (multi state) in the following form: topic: input file
    Home/Hall/Temperature="/sys/bus/w1/devices/28-000004a7b5bb/w1_slave"

[Raspberry Pi 2 - Output Sensors Mapping]

    [[Two State - Output list]]
    #List of output sensors (two state) in the following form: topic:GPIO pin
        number
    Home/Hall/Heater_Switch=24
    Home/LivingRoom/Light_Switch=23
```

A.4 mySQL_graphs_client.py

```
#!/usr/bin/python
from configobj import ConfigObj, ConfigObjError
import sys, os, argparse, atexit, logging, grp, pwd, getpass, json, time, MySQLdb
import paho.mqtt.client as mqtt
from signal import SIGTERM
import pandas as pd
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import matplotlib.dates as dates
```

```
11
12 execfile("./mySQL_graphs_helper.py")
13
14 loggingFilename="./mySQL_graphs_client.log"
15 logging.basicConfig(filename=loggingFilename,format='%(asctime)s %(message)s',
       datefmt='%d/%m/%Y %I:%M:%S %p',level=logging.DEBUG)
16 logging.info("Started")
17
18 logging.info("Reading Configuration File...")
19 cfgFilename="./mysql_client.cfg"
20
21 try:
22     config = ConfigObj(cfgFilename,unrepr=True, file_error=True)
23 except (ConfigObjError, IOError), e:
24     logging.error( 'Could not read "%s": %s' % (cfgFilename, e))
25     exit()
26
27 clientId = config['Mqtt Client Details']['ClientId']
28 server = config['Mqtt Client Details']['Server']
29 port = config['Mqtt Client Details']['Port']
30 cleanSession = config['Mqtt Client Details']['Clean Session']
31 timeOut = config['Mqtt Client Details']['Time Out']
32 keepAliveTimeout = config['Mqtt Client Details']['Keep Alive Timeout']
33 username = config['Mqtt Client Details']['UserName']
34 password = config['Mqtt Client Details']['Password']
35 SSL = config['Mqtt Client Details']['SSL']
36 ca_certs_path = config['Mqtt Client Details']['CA Cert Path']
37
38 mysql_server = config['MySql Details']['Server']
39 #mysql_port = config['MySql Detail']['Port']
40 mysql_username = config['MySql Details']['Username']
41 mysql_passwd  = config['MySql Details']['Password']
42 mysql_db  = config['MySql Details']['Database']
43 twoStateSensorList = config['Other']['TwoStateSensorList']
44 multiStateSensorList = config['Other']['MultiStateSensorList']
45 commands_topic = "Home/Commands/Graph"
46
47 subscribe_list=twoStateSensorList.split(",")+multiStateSensorList.split(",")
48 subscribe_list.append(commands_topic)
49
50 logging.info("Setting up MySql Database...")
51
52 try:
53     # Open database connection
54     db = MySQLdb.connect(mysql_server,mysql_username, mysql_passwd, mysql_db)
55     # prepare a cursor object using cursor() method
56     mysql_c = db.cursor()
57 except MySQLdb.Error, e:
58     logging.error('MySQLdb connection error: $s ' % e )
59     exit()
60
61 logging.info("Setting up MQTT client...")
62 myData = None
63 client = mqtt.Client(clientId,cleanSession,myData,"MQTTv311")
64 client.on_connect = on_connect
65 client.on_message = on_message
66 client.on_publish = on_publish
67 #client.on_subscribe = on_subscribe
68 client.max_inflight_messages_set(20)
69 client.message_retry_set(timeOut)
70
71 if SSL:
72     client.tls_set(ca_certs_path,None,None,ssl.CERTR_REQUIRED,ss1.PROTOCOL_TLSv1,
          None)
73
74     if username!=None:
75     client.username_pw_set(username,password)
76
77 client.user_data_set(myData)
78 logging.info("Connecting to broker...")
79 client.connect(server,port,keepAliveTimeout)
80 client.loop_forever()
```

A.5 mySQL_graphs_helper.py

```
1
2 # The callback for when the client receives a CONNACK response from the server.
3 def on_connect(client, userdata, flags, rc):
4     logging.info("Connected with result code "+str(rc))
5
6     for i in subscribe_list:
7         logging.info("Subscribing to topic: %s" % i)
8         client.subscribe(i)
9
10 # The callback for when a PUBLISH message is received from the server.
11 def on_message(client, userdata, msg):
12     logging.info("Message received on topic: %s with payload: %s." % (msg.topic,
          msg.payload))
13     if not msg.retain:
14         if (not msg.topic==commands_topic):
15             insert2mysql(client,userdata,msg)
16         elif not msg.payload=="Ready":
17             draw_graph(client,userdata,msg)
18
19 def draw_graph(client,userdata,msg):
20     logging.info("Drawing a graph from mySql database")
21
22     components=msg.payload.split(",")
23     table_name=components[0]
24     table_name = table_name.replace("/", "_")
25     startDate=components[1]
26     endDate=components[2]
27     query= "SELECT Time, State FROM "+table_name + " Where Time BETWEEN '"+
          startDate + " 00:00:00' AND '" + endDate + " 23:59:59'"
28     df = pd.read_sql(query,db,index_col=['Time'])
29     if df.empty:
30         plt.yticks([])
31         plt.xticks([])
32         plt.title("No Values To Show")
33         plt.plot(None,None)
34         plt.savefig('/var/www/html/viewGraph/myGraph.png')
35         plt.close()
36     elif components[0] in twoStateSensorList:
37         ax=df.plot(drawstyle="steps-post", linewidth=2,ylim=(0,1.5))
38         ax.set_yticks((0,1))
39         ax.set_yticklabels(('OFF','ON'))
40         ax.set_xlabel("")
41         ax.set_title(table_name)
42         ax.xaxis.set_minor_locator(dates.AutoDateLocator())
43         ax.xaxis.set_minor_formatter(dates.DateFormatter('%HH'))
44         ax.xaxis.set_major_locator(dates.DayLocator())
45         ax.xaxis.set_major_formatter(dates.DateFormatter('\n\n\n%d\n%a'))
46         fig = ax.get_figure()
47         fig.savefig('/var/www/html/viewGraph/myGraph.png')
48         plt.close(fig)
49     elif components[0] in multiStateSensorList:
50         ax=df.plot(drawstyle="line", linewidth=2)
51         ax.set_xlabel("")
52         ax.set_title(table_name)
53         ax.xaxis.set_minor_locator(dates.AutoDateLocator())
54         ax.xaxis.set_minor_formatter(dates.DateFormatter('%HH'))
55         ax.xaxis.set_major_locator(dates.DayLocator())
56         ax.xaxis.set_major_formatter(dates.DateFormatter('\n\n\n%d\n%a'))
57         fig = ax.get_figure()
58         fig.savefig('/var/www/html/viewGraph/myGraph.png')
59         plt.close(fig)
60
61     client.publish("Home/Commands/Graph","Ready",qos=2,retain=False)
62
63     logging.info("Draw_graph finished")
64
65 def insert2mysql(client,userdata,msg):
66     logging.info("Insert new Data to mySql database")
67
68     component=msg.payload.split('@')
```

```
69      message=component[0]
70      date=component[1]
71      if (message=="ON"):
72          value=1
73      elif (message=="OFF"):
74          value=0
75      else:
76          value = message.split('*')[0]
77
78
79
80      table_name = msg.topic.replace("/", "_")
81      query1="CREATE TABLE IF NOT EXISTS %s (Id INT PRIMARY KEY AUTO_INCREMENT, Time
            DATETIME ,State INT)" % table_name
82      result=mysql_c.execute(query1)
83      db.commit()
84      query2=" INSERT INTO "+ table_name + "(Time,State) VALUES(%s,%s)"
85      result=mysql_c.execute(query2,(date,value))
86      db.commit()
87
88      logging.info("Insert2mysql finished")
89
90  def on_publish(client,userdata,mid):
91      logging.info("Message with id "+str(mid)+" delivered")
```

A.6 mySQL_graphs_client.cfg

```
1  [Mqtt Client Details]
2
3  ClientId="DatabaseClient"
4  Server="localhost"
5  Port=1883
6  Clean Session=True
7  #Optional
8  Time Out=5
9  Keep Alive Timeout=200
10 UserName="myname"
11 Password="mypass"
12 SSL=False
13 CA Cert Path=
14
15 [MySql Details]
16
17 Server="localhost"
18 Port=
19 Username="root"
20 Password=""
21 Database="Sensors"
22
23 [Other]
24
25 TwoStateSensorList= "Home/Hall/Window_1,Home/Hall/Window_2,Home/Hall/Heater_Switch
       ,Home/LivingRoom/Light_Switch"
26 MultiStateSensorList="Home/Hall/Temperature"
```

A.7 sensors_simulation.py

```
1  import RPi.GPIO as GPIO
2  import random
3  from time import sleep
4  from configobj import ConfigObj, ConfigObjError
5
6  X=150
7
8  cfgFilename="./sensors_client.cfg"
9  try:
10     config = ConfigObj(cfgFilename,unrepr=True, file_error=True)
11 except  (ConfigObjError, IOError), e:
12     logging.error( 'Could not read "%s": %s' % (cfgFilename, e))
13
14 GPIO.setmode(GPIO.BCM)
```

```
GPIO.setwarnings(False)

inputMap = config['Raspberry Pi 2 - Input Sensors Mapping']['Two State - Input
    list']
inputPins= inputMap.values()

GPIO.setup(inputPins,GPIO.OUT)

on_off=[0,1]
timeBag=[0.1,1,2,3,4,5,7,9,13,16,20,25]
try:
    while True:
        random_pin=random.choice(inputPins)
        to_do=random.choice(on_off)
        GPIO.output(random_pin,to_do)
        sleep(random.choice(timeBag)*X)

except KeyboardInterrupt:
    GPIO.cleanup()
```

A.8 graph.html

```
<!--
    Chrysostomos Christou, 2016, NTUA, Thisis
This file must be in directory /var/www/html/viewGraph/
-->
<html>
<body>
<meta name="viewport" content="width=device-width, initial-scale=1" />
<img src="myGraph.png" alt="My Graph" style="max-width:100%;height:auto;"></img>
</body>
</html>
```

B Hm-Monitoring

Ακολουθούν οι βασικές κλάσεις για την υλοποίηση της εφαρμογής Android.

B.1 ClientConnections.java

```
//Original work
/*******************************************************************************
 * Copyright (c) 1999, 2014 IBM Corp.
 *
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * and Eclipse Distribution License v1.0 which accompany this distribution.
 *
 * The Eclipse Public License is available at
 *    http://www.eclipse.org/legal/epl-v10.html
 * and the Eclipse Distribution License is available at
 *   http://www.eclipse.org/org/documents/edl-v10.php.
 */
//Modified work
/*
 * Chrysostomos Christou, 2016, Thesis
 */
package com.chrysostomos.homemonitoring;

import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v4.content.ContextCompat;
import android.util.Log;
import android.view.ActionMode;
```

```
30 import android.view.Menu;
31 import android.view.MenuInflater;
32 import android.view.MenuItem;
33 import android.view.MenuItem.OnMenuItemClickListener;
34 import android.view.View;
35 import android.widget.AdapterView;
36 import android.widget.AdapterView.OnItemLongClickListener;
37 import android.widget.ArrayAdapter;
38 import android.widget.ListView;
39 import android.widget.Toast;
40
41 import org.eclipse.paho.android.service.MqttAndroidClient;
42 import com.chrysostomos.homemonitoring.Connection.ConnectionStatus;
43
44 import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
45 import org.eclipse.paho.client.mqttv3.MqttException;
46 import org.eclipse.paho.client.mqttv3.MqttSecurityException;
47
48 import java.beans.PropertyChangeEvent;
49 import java.beans.PropertyChangeListener;
50 import java.io.FileInputStream;
51 import java.io.FileNotFoundException;
52 import java.util.Map;
53
54 /**
55  * ClientConnections is the main activity for the homemonitoring application, it
56  * displays all the active connections.
57  *
58  */
59 public class ClientConnections extends ListActivity {
60
61   /**
62    * Token to pass to the MQTT Service
63    */
64   final static String TOKEN = "com.chrysostomos.homemonitoring.ClientConnections";
65
66   /**
67    * ArrayAdapter to populate the list view
68    */
69   private ArrayAdapter<Connection> arrayAdapter = null;
70
71   /**
72    * {@link ChangeListener} for use with all {@link Connection} objects created by
       this instance of <code>ClientConnections</code>
73    */
74   private ChangeListener changeListener = new ChangeListener();
75
76   /**
77    * This instance of <code>ClientConnections</code> used to update the UI in {
       @link ChangeListener}
78    */
79   private ClientConnections clientConnections = this;
80
81   /**
82    * Contextual action bar active or not
83    */
84   private boolean contextualActionBarActive = false;
85
86   /**
87    * @see android.app.ListActivity#onCreate(Bundle)
88    */
89   @Override
90   protected void onCreate(Bundle savedInstanceState) {
91     super.onCreate(savedInstanceState);
92
93     ListView connectionList = getListView();
94     View header = getLayoutInflater().inflate(R.layout.client_connections_header,
           null);
95     connectionList.addHeaderView(header, null, false);
96     connectionList.setOnItemLongClickListener(new LongClickItemListener());
97     connectionList.setTextFilterEnabled(true);
98     arrayAdapter = new ArrayAdapter<Connection>(this,
99         R.layout.connection_text_view);
```

```
100     setListAdapter(arrayAdapter);
101
102     // get all the available connections
103     Map<String, Connection> connections = Connections.getInstance(this)
104         .getConnections();
105
106     if (connections != null) {
107       for (String s : connections.keySet())
108       {
109         arrayAdapter.add(connections.get(s));
110       }
111     }
112
113   }
114
115   /**
116    * Creates the action bar for the activity
117    *
118    * @see ListActivity#onCreateOptionsMenu(Menu)
119    */
120   @Override
121   public boolean onCreateOptionsMenu(Menu menu) {
122
123     OnMenuItemClickListener menuItemClickListener = new Listener(this);
124
125     getMenuInflater().inflate(R.menu.activity_connections, menu);
126     menu.findItem(R.id.newConnection).setOnMenuItemClickListener(
127         menuItemClickListener);
128
129     return true;
130   }
131
132   /**
133    * Listens for item clicks on the view
134    *
135    * @param listView
136    *            The list view where the click originated from
137    * @param view
138    *            The view which was clicked
139    * @param position
140    *            The position in the list that was clicked
141    */
142   @Override
143   protected void onListItemClick(ListView listView, View view, int position,
144       long id) {
145     super.onListItemClick(listView, view, position, id);
146
147     if (!contextualActionBarActive) {
148       Connection c =(Connection) listView.getItemAtPosition(position);
149
150       // start the connectionDetails activity to display the details about the
151       // selected connection
152       Intent intent = new Intent();
153       intent.setClassName(getApplicationContext().getPackageName(),
154           "com.chrysostomos.homemonitoring.ConnectionDetails");
155       intent.putExtra("handle", c.handle());
156       startActivity(intent);
157     }
158
159   }
160
161   /**
162    * @see ListActivity#onActivityResult(int,int,Intent)
163    */
164   @Override
165   protected void onActivityResult(int requestCode, int resultCode, Intent data) {
166
167     if (resultCode == RESULT_CANCELED) {
168       return;
169     }
170
171     Bundle dataBundle = data.getExtras();
172
```

```
173     // perform connection create and connect
174     connectAction(dataBundle);
175
176   }
177
178   /**
179    * @see ListActivity#onResume()
180    */
181   @Override
182   protected void onResume() {
183     super.onResume();
184     arrayAdapter.notifyDataSetChanged();
185
186     //Recover connections.
187     Map<String, Connection> connections = Connections.getInstance(this).
           getConnections();
188
189     //Register receivers again
190     for (Connection connection : connections.values()){
191       connection.getClient().registerResources(this);
192       connection.getClient().setCallback(new MqttCallbackHandler(this, connection.
             getClient().getServerURI()+connection.getClient().getClientId()));
193     }
194   }
195
196   /**
197    * @see ListActivity#onDestroy()
198    */
199   @Override
200   protected void onDestroy() {
201
202     Map<String, Connection> connections = Connections.getInstance(this).
           getConnections();
203
204     for (Connection connection : connections.values()){
205       connection.registerChangeListener(changeListener);
206       connection.getClient().unregisterResources();
207     }
208     super.onDestroy();
209   }
210
211   /**
212    * Process data from the connect action
213    *
214    * @param data the {@link Bundle} returned by the {@link NewConnection} Acitivty
215    */
216   private void connectAction(Bundle data) {
217     MqttConnectOptions conOpt = new MqttConnectOptions();
218     /*
219      * Mutal Auth connections could do something like this
220      *
221      *
222      * SSLContext context = SSLContext.getDefault();
223      * context.init({new CustomX509KeyManager()},null,null); //where
             CustomX509KeyManager proxies calls to keychain api
224      * SSLSocketFactory factory = context.getSSLSocketFactory();
225      *
226      * MqttConnectOptions options = new MqttConnectOptions();
227      * options.setSocketFactory(factory);
228      *
229      * client.connect(options);
230      *
231      */
232
233     // The basic client information
234     String server = (String) data.get(ActivityConstants.server);
235     String clientId = (String) data.get(ActivityConstants.clientId);
236     int port = Integer.parseInt((String) data.get(ActivityConstants.port));
237     boolean ssl = (Boolean) data.get(ActivityConstants.ssl);
238     String ssl_key = (String) data.get(ActivityConstants.ssl_key);
239     String ssl_pass = (String) data.get(ActivityConstants.ssl_pass);
240     String uri = null;
241     if (ssl) {
```

```
242         Log.e("SSLConnection", "Doing an SSL Connect");
243         uri = "ssl://";
244
245       }
246       else {
247         uri = "tcp://";
248       }
249
250       uri = uri + server + ":" + port;
251
252       MqttAndroidClient client;
253       client = Connections.getInstance(this).createClient(this, uri, clientId);
254
255       if (ssl){
256           try {
257               if(ssl_key != null && !ssl_key.equalsIgnoreCase(""))
258               {
259                   FileInputStream key = new FileInputStream(ssl_key);
260                   conOpt.setSocketFactory(client.getSSLSocketFactory(key,
261                           ssl_pass));
262                 SharedPreferences myPrefs = PreferenceManager.
                      getDefaultSharedPreferences(this);
263                 SharedPreferences.Editor myPrefsEditor;
264                 myPrefsEditor= myPrefs.edit();
265                 myPrefsEditor.putString("ssl_key", ssl_key);
266                 Log.d("ClientConn", "SSL_key:  : " + ssl_key);
267                 myPrefsEditor.putString("ssl_pass", ssl_pass);
268                 Log.d("ClientConn", "SSL_pass:  : " + ssl_pass);
269                 myPrefsEditor.commit();
270               }
271
272           } catch (MqttSecurityException e) {
273               Log.e(this.getClass().getCanonicalName(),
274                       "MqttException Occured: ", e);
275           } catch (FileNotFoundException e) {
276               Log.e(this.getClass().getCanonicalName(),
277                       "MqttException Occured: SSL Key file not found", e);
278           }
279       }
280
281       // create a client handle
282       String clientHandle = uri + clientId;
283
284       // connection options
285       String username = (String) data.get(ActivityConstants.username);
286       String password = (String) data.get(ActivityConstants.password);
287
288       boolean cleanSession = ActivityConstants.defaultCleanSession;
289       int timeout = ActivityConstants.defaultTimeOut;
290       int keepalive =  ActivityConstants.defaultKeepAlive;
291
292       Connection connection = new Connection(clientHandle, clientId, server, port,
293           this, client, ssl);
294       arrayAdapter.add(connection);
295
296       connection.registerChangeListener(changeListener);
297       // connect client
298
299       String[] actionArgs = new String[1];
300       actionArgs[0] = clientId;
301       connection.changeConnectionStatus(ConnectionStatus.CONNECTING);
302
303       conOpt.setCleanSession(cleanSession);
304       conOpt.setConnectionTimeout(timeout);
305       conOpt.setKeepAliveInterval(keepalive);
306       if (!username.equals(ActivityConstants.empty)) {
307         conOpt.setUserName(username);
308       }
309       if (!password.equals(ActivityConstants.empty)) {
310         conOpt.setPassword(password.toCharArray());
311       }
312
313       final ActionListener callback = new ActionListener(this,
```

```
314         ActionListener.Action.CONNECT, clientHandle, actionArgs);
315
316
317     client.setCallback(new MqttCallbackHandler(this, clientHandle));
318
319
320     //set traceCallback
321     client.setTraceCallback(new MqttTraceCallback());
322
323     connection.addConnectionOptions(conOpt);
324     Connections.getInstance(this).addConnection(connection);
325
326     try {
327         client.connect(conOpt, null, callback);
328       }
329     catch (MqttException e) {
330         Log.e(this.getClass().getCanonicalName(),
331             "MqttException Occured", e);
332     }
333
334
335   }
336
337   /**
338    * <code>LongClickItemListener</code> deals with enabling and disabling the
          contextual action bar and
339    * processing the actions selected.
340    *
341    */
342   private class LongClickItemListener implements OnItemLongClickListener,
          ActionMode.Callback, OnClickListener {
343
344     /** The index of the item selected, or -1 if an item is not selected **/
345     private int selected = -1;
346     /** The view of the item selected **/
347     private View selectedView = null;
348     /** The connection the view is representing **/
349     private Connection connection = null;
350
351     /* (non-Javadoc)
352      * @see android.widget.AdapterView.OnItemLongClickListener#onItemLongClick(
              android.widget.AdapterView, android.view.View, int, long)
353      */
354     @Override
355     public boolean onItemLongClick(AdapterView<?> parent, View view, int position,
            long id) {
356       clientConnections.startActionMode(this);
357       selected = position;
358       selectedView = view;
359       view.setSelected(true);
360       clientConnections.getListView().setSelection(position);
361       connection = (Connection) clientConnections.getListView().getItemAtPosition(
              position);
362       selectedView.setBackgroundColor(ContextCompat.getColor(getApplicationContext
              (), android.R.color.holo_blue_dark));
363       return true;
364     }
365
366     /* (non-Javadoc)
367      * @see android.view.ActionMode.Callback#onActionItemClicked(android.view.
              ActionMode, android.view.MenuItem)
368      */
369     @Override
370     public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
371       switch (item.getItemId()) {
372         case R.id.delete :
373           delete();
374           mode.finish();
375           return true;
376         default :
377           return false;
378       }
379     }
```

```
380
381     /* (non-Javadoc)
382      * @see android.view.ActionMode.Callback#onCreateActionMode(android.view.
             ActionMode, android.view.Menu)
383      */
384     @Override
385     public boolean onCreateActionMode(ActionMode mode, Menu menu) {
386       MenuInflater inflater = mode.getMenuInflater();
387       inflater.inflate(R.menu.activity_client_connections_contextual, menu);
388       clientConnections.contextualActionBarActive = true;
389       getListView().setClickable(false);
390       return true;
391     }
392
393     /* (non-Javadoc)
394      * @see android.view.ActionMode.Callback#onDestroyActionMode(android.view.
             ActionMode)
395      */
396     @Override
397     public void onDestroyActionMode(ActionMode mode) {
398       selected = -1;
399       selectedView.setBackgroundColor(ContextCompat.getColor(getApplicationContext
              (), android.R.color.transparent));
400       selectedView = null;
401       clientConnections.contextualActionBarActive = false;
402       getListView().setClickable(true);
403     }
404
405     /* (non-Javadoc)
406      * @see android.view.ActionMode.Callback#onPrepareActionMode(android.view.
             ActionMode, android.view.Menu)
407      */
408     @Override
409     public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
410       return false;
411     }
412
413     /**
414      * Deletes the connection, disconnecting if required.
415      */
416     private void delete()
417     {
418
419       if (connection.isConnectedOrConnecting()) {
420         (new Notify()).toast(clientConnections, "Please disconnect client first",
              Toast.LENGTH_SHORT);
421       }
422       else {
423         SharedPreferences myPrefs = getSharedPreferences(getPackageName(),
              getApplicationContext().MODE_PRIVATE);
424         SharedPreferences.Editor myPrefsEditor =myPrefs.edit();
425         myPrefsEditor.remove(connection.handle() + ":" + "topicListJson");
426         myPrefsEditor.commit();
427         myPrefsEditor.remove(connection.handle() + ":" + "myRoomsJson");
428         myPrefsEditor.commit();
429         myPrefsEditor.remove(connection.handle() + ":" + "roomListJson");
430         myPrefsEditor.commit();
431
432         arrayAdapter.remove(connection);
433         Connections.getInstance(clientConnections).removeConnection(connection);
434         Log.d("Cl.Connections:delete","connection deleted");
435       }
436
437     }
438
439     @Override
440     public void onClick(DialogInterface dialog, int which) {
441       //user pressed continue disconnect client and delete
442       try {
443         connection.getClient().disconnect();
444       }
445       catch (MqttException e) {
446         e.printStackTrace();
```

```
447         }
448         arrayAdapter.remove(connection);
449         Connections.getInstance(clientConnections).removeConnection(connection);
450 
451       }
452     }
453 
454     /**
455      * This class ensures that the user interface is updated as the Connection
           objects change their states
456      *
457      *
458      */
459     private class ChangeListener implements PropertyChangeListener {
460 
461       /**
462        * @see java.beans.PropertyChangeListener#propertyChange(java.beans.
             PropertyChangeEvent)
463        */
464       @Override
465       public void propertyChange(PropertyChangeEvent event) {
466 
467         if (!event.getPropertyName().equals(ActivityConstants.
               ConnectionStatusProperty)) {
468           return;
469         }
470         clientConnections.runOnUiThread(new Runnable() {
471 
472           @Override
473           public void run() {
474             Log.d("ClientConnections","propertyChange");
475             clientConnections.arrayAdapter.notifyDataSetChanged();
476           }
477 
478         });
479 
480       }
481 
482     }
483 }
```

B.2 NewConnection.java

```
1  /*
2   * Chrysostomos Christou, 2016, Thesis
3   */
4  package com.chrysostomos.homemonitoring;
5 
6  import android.app.Activity;
7  import android.app.Dialog;
8  import android.content.Intent;
9  import android.os.Bundle;
10 import android.support.v4.app.NavUtils;
11 import android.view.Menu;
12 import android.view.MenuItem;
13 import android.view.MenuItem.OnMenuItemClickListener;
14 import android.view.View;
15 import android.widget.ArrayAdapter;
16 import android.widget.AutoCompleteTextView;
17 import android.widget.Button;
18 import android.widget.CheckBox;
19 import android.widget.EditText;
20 import android.widget.Toast;
21 
22 import java.io.BufferedReader;
23 import java.io.BufferedWriter;
24 import java.io.File;
25 import java.io.FileReader;
26 import java.io.FileWriter;
27 import java.io.IOException;
28 import java.util.ArrayList;
29 import java.util.HashMap;
```

```
30 import java.util.Map;
31
32 /**
33  * Handles collection of user information to create a new MQTT Client
34  *
35  */
36 public class NewConnection extends Activity {
37
38
39   private int openfileDialogId = 0;
40   /**
41    * @see android.app.Activity#onCreate(android.os.Bundle)
42    */
43   @Override
44   protected void onCreate(Bundle savedInstanceState) {
45     super.onCreate(savedInstanceState);
46     setContentView(R.layout.activity_new_connection);
47
48     //load auto compete options
49
50     ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout
           .simple_list_item_1);
51     adapter.addAll(readHosts());
52     AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.
           serverURI);
53     textView.setAdapter(adapter);
54
55     ((Button) findViewById(R.id.sslKeyBut)).setOnClickListener(new View.
           OnClickListener() {
56
57       @Override
58       public void onClick(View v) {
59         //showFileChooser();
60         showDialog(openfileDialogId);
61       }
62     });
63
64     ((CheckBox) findViewById(R.id.sslCheckBox)).setOnClickListener(new View.
           OnClickListener() {
65
66       @Override
67       public void onClick(View v) {
68         if (((CheckBox) v).isChecked()) {
69           ((Button) findViewById(R.id.sslKeyBut)).setClickable(true);
70           ((EditText) findViewById(R.id.sslPassword)).setEnabled(true);
71         } else {
72           ((Button) findViewById(R.id.sslKeyBut)).setClickable(false);
73           ((EditText) findViewById(R.id.sslPassword)).setEnabled(false);
74         }
75
76       }
77     });
78
79     ((Button)findViewById(R.id.sslKeyBut)).setClickable(false);
80
81   }
82
83   /**
84    * @see android.app.Activity#onCreateOptionsMenu(android.view.Menu)
85    */
86   @Override
87   public boolean onCreateOptionsMenu(Menu menu) {
88     getMenuInflater().inflate(R.menu.activity_new_connection, menu);
89     OnMenuItemClickListener listener = new Listener(this);
90     menu.findItem(R.id.connectAction).setOnMenuItemClickListener(listener);
91     return true;
92   }
93
94   /**
95    * @see android.app.Activity#onOptionsItemSelected(android.view.MenuItem)
96    */
97   @Override
98   public boolean onOptionsItemSelected(MenuItem item) {
```

```
99      switch (item.getItemId()) {
100       case android.R.id.home :
101         NavUtils.navigateUpFromSameTask(this);
102         return true;
103     }
104     return super.onOptionsItemSelected(item);
105   }
106
107
108   /**
109    * Handles action bar actions
110    *
111    */
112   private class Listener implements OnMenuItemClickListener {
113
114     //used for starting activities
115     private NewConnection newConnection = null;
116
117     public Listener(NewConnection newConnection)
118     {
119       this.newConnection = newConnection;
120     }
121
122     /**
123      * @see android.view.MenuItem.OnMenuItemClickListener#onMenuItemClick(android.
           view.MenuItem)
124      */
125     @Override
126     public boolean onMenuItemClick(MenuItem item) {
127       {
128         // this will only connect need to package up and sent back
129
130         int id = item.getItemId();
131
132         Intent dataBundle = new Intent();
133
134         switch (id) {
135           case R.id.connectAction :
136             //extract client information
137             String server = ((AutoCompleteTextView) findViewById(R.id.serverURI))
138                 .getText().toString();
139             String port = ((EditText) findViewById(R.id.port))
140                 .getText().toString();
141             String clientId = ((EditText) findViewById(R.id.clientId))
142                 .getText().toString();
143
144             if (server.equals(ActivityConstants.empty) || port.equals(
                    ActivityConstants.empty) || clientId.equals(ActivityConstants.
                    empty))
145             {
146               String notificationText = newConnection.getString(R.string.
                      missingOptions);
147               Notify.toast(newConnection, notificationText, Toast.LENGTH_LONG);
148               return false;
149             }
150
151
152             // get all advance options
153             String username = ((EditText) findViewById(R.id.uname)).getText()
154                     .toString();
155             String password = ((EditText) findViewById(R.id.password))
156                     .getText().toString();
157             String sslkey = null;
158             String sslPass = null;
159             boolean ssl = ((CheckBox) findViewById(R.id.sslCheckBox)).isChecked();
160             if(ssl)
161             {
162               sslkey = ((EditText) findViewById(R.id.sslKeyLocaltion))
163                       .getText().toString();
164               sslPass = ((EditText) findViewById(R.id.sslPassword))
165                       .getText().toString();
166
167             }
```

```
168
169             //persist server
170             persistServerURI(server);
171
172             //put data into a bundle to be passed back to ClientConnections
173             dataBundle.putExtra(ActivityConstants.server, server);
174             dataBundle.putExtra(ActivityConstants.port, port);
175             dataBundle.putExtra(ActivityConstants.clientId, clientId);
176             dataBundle.putExtra(ActivityConstants.action, ActivityConstants.
                    connect);
177             dataBundle.putExtra(ActivityConstants.username, username);
178             dataBundle.putExtra(ActivityConstants.password, password);
179             dataBundle.putExtra(ActivityConstants.ssl, ssl);
180             dataBundle.putExtra(ActivityConstants.ssl_key, sslkey);
181             dataBundle.putExtra(ActivityConstants.ssl_pass, sslPass);
182
183             setResult(RESULT_OK, dataBundle);
184             newConnection.finish();
185             break;
186           }
187           return false;
188
189         }
190
191       }
192
193       /**
194        * Add a server URI to the persisted file
195        *
196        * @param serverURI the uri to store
197        */
198       private void persistServerURI(String serverURI) {
199         File fileDir = newConnection.getFilesDir();
200         File presited = new File(fileDir, "hosts.txt");
201         BufferedWriter bfw = null;
202         try {
203           bfw = new BufferedWriter(new FileWriter(presited));
204           bfw.write(serverURI);
205           bfw.newLine();
206         }
207         catch (IOException e) {
208           // TODO Auto-generated catch block
209           e.printStackTrace();
210         }
211         finally {
212           try {
213             if (bfw != null) {
214               bfw.close();
215             }
216           }
217           catch (IOException e) {
218             // TODO Auto-generated catch block
219             e.printStackTrace();
220           }
221         }
222       }
223
224     }
225
226     /**
227      * Read persisted hosts
228      * @return The hosts contained in the persisted file
229      */
230     private String[] readHosts() {
231       File fileDir = getFilesDir();
232       File persisted = new File(fileDir, "hosts.txt");
233       if (!persisted.exists()) {
234         return new String[0];
235       }
236       ArrayList<String> hosts = new ArrayList<String>();
237       BufferedReader br = null;
238       try {
239         br = new BufferedReader(new FileReader(persisted));
```

```
240         String line = null;
241         line = br.readLine();
242         while (line != null) {
243           hosts.add(line);
244           line = br.readLine();
245         }
246       }
247       catch (IOException e) {
248         e.printStackTrace();
249       }
250       finally {
251         try {
252           if (br != null) {
253             br.close();
254           }
255         }
256         catch (IOException e) {
257           // TODO Auto-generated catch block
258           e.printStackTrace();
259         }
260       }
261
262       return hosts.toArray(new String[hosts.size()]);
263
264     }
265
266     @Override
267     protected Dialog onCreateDialog(int id) {
268       if (id == openfileDialogId) {
269         Map<String, Integer> images = new HashMap<String, Integer>();
270         images.put(OpenFileDialog.sRoot, R.drawable.ic_launcher);
271         images.put(OpenFileDialog.sParent, R.drawable.ic_launcher);
272         images.put(OpenFileDialog.sFolder, R.drawable.ic_launcher);
273         images.put("bks", R.drawable.ic_launcher);
274         images.put(OpenFileDialog.sEmpty, R.drawable.ic_launcher);
275         Dialog dialog = OpenFileDialog.createDialog(id, this, "openfile",
276                 new CallbackBundle() {
277                   @Override
278                   public void callback(Bundle bundle) {
279                     String filepath = bundle.getString("path");
280                     // setTitle(filepath);
281                     ((EditText) findViewById(R.id.sslKeyLocaltion))
282                             .setText(filepath);
283                   }
284                 }, ".bks;", images);
285         return dialog;
286       }
287       return null;
288     }
289 }
```

B.3 ConnectionDetails.java

```
1  //Original work
2  /*******************************************************************************
3   * Copyright (c) 1999, 2014 IBM Corp.
4   *
5   * All rights reserved. This program and the accompanying materials
6   * are made available under the terms of the Eclipse Public License v1.0
7   * and Eclipse Distribution License v1.0 which accompany this distribution.
8   *
9   * The Eclipse Public License is available at
10  *    http://www.eclipse.org/legal/epl-v10.html
11  * and the Eclipse Distribution License is available at
12  *   http://www.eclipse.org/org/documents/edl-v10.php.
13  */
14 //Modified work
15 /*
16  * Chrysostomos Christou, 2016, Thesis
17  */
18 package com.chrysostomos.homemonitoring;
19
```

```
20 import android.app.ActionBar;
21 import android.app.FragmentTransaction;
22 import android.content.SharedPreferences;
23 import android.os.Bundle;
24 import android.support.v4.app.Fragment;
25 import android.support.v4.app.FragmentActivity;
26 import android.support.v4.app.FragmentManager;
27 import android.support.v4.app.FragmentPagerAdapter;
28 import android.support.v4.view.ViewPager;
29 import android.util.Log;
30 import android.view.Menu;
31 import android.view.ViewGroup;
32
33 import com.google.gson.Gson;
34 import com.google.gson.reflect.TypeToken;
35
36 import java.beans.PropertyChangeEvent;
37 import java.beans.PropertyChangeListener;
38 import java.lang.reflect.Type;
39 import java.util.ArrayList;
40 import java.util.HashMap;
41
42 /**
43  * The connection details activity operates the fragments that make up the
44  * connection details screen.
45  * <p>
46  * The fragments which this FragmentActivity uses are
47  * <ul>
48  * <li>{@link LogFragment}
49  * <li>{@link DisplaySensorsFragment}
50  * <li>{@link InsertDeviceFragment}
51  * </ul>
52  *
53  */
54 public class ConnectionDetails extends FragmentActivity implements
55     ActionBar.TabListener {
56
57   public HashMap<String,String> topicList = new HashMap<>();
58   private SharedPreferences myPrefs;
59   SharedPreferences.Editor myPrefsEditor;
60   Gson gson = new Gson();
61
62   private static final String TAG_DISPLAY_FRAGMENT = "display_fragment";
63   /**
64    * {@link SectionsPagerAdapter} that is used to get pages to display
65    */
66   SectionsPagerAdapter sectionsPagerAdapter;
67   /**
68    * {@link ViewPager} object allows pages to be flipped left and right
69    */
70   ViewPager viewPager;
71
72   /** The currently selected tab **/
73   private int selected = 0;
74
75   /**
76    * The handle to the {@link Connection} which holds the data for the client
77    * selected
78    **/
79   private String clientHandle = null;
80
81
82
83   /** This instance of <code>ConnectionDetails</code> **/
84   private final ConnectionDetails connectionDetails = this;
85
86   /**
87    * The instance of {@link Connection} that the <code>clientHandle</code>
88    * represents
89    **/
90   private Connection connection = null;
91
92   /**
```

```
93      * The {@link ChangeListener} this object is using for the connection
94      * updates
95      **/
96     private ChangeListener changeListener = null;
97
98     /**
99      * @see android.support.v4.app.FragmentActivity#onCreate(android.os.Bundle)
100     */
101    @Override
102    protected void onCreate(Bundle savedInstanceState) {
103      super.onCreate(savedInstanceState);
104
105      clientHandle = getIntent().getStringExtra("handle");
106      setContentView(R.layout.activity_connection_details);
107
108      // Create the adapter that will return a fragment for each of the pages
109      sectionsPagerAdapter = new SectionsPagerAdapter(
110          getSupportFragmentManager());
111
112      // Set up the action bar for tab navigation
113      final ActionBar actionBar = getActionBar();
114      actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
115
116      // add the sectionsPagerAdapter
117      viewPager = (ViewPager) findViewById(R.id.pager);
118      viewPager.setAdapter(sectionsPagerAdapter);
119
120      viewPager
121          .setOnPageChangeListener(new ViewPager.SimpleOnPageChangeListener() {
122
123            @Override
124            public void onPageSelected(int position) {
125              // select the tab that represents the current page
126              actionBar.setSelectedNavigationItem(position);
127
128            }
129          });
130
131      // Create the tabs for the screen
132      for (int i = 0; i < sectionsPagerAdapter.getCount(); i++) {
133        ActionBar.Tab tab = actionBar.newTab();
134        tab.setText(sectionsPagerAdapter.getPageTitle(i));
135        tab.setTabListener(this);
136        actionBar.addTab(tab);
137      }
138
139      connection = Connections.getInstance(this).getConnection(clientHandle);
140      setTitle(connection.getId());
141      changeListener = new ChangeListener();
142      connection.registerChangeListener(changeListener);
143
144      viewPager.setOffscreenPageLimit(sectionsPagerAdapter.getCount());
145
146      myPrefs = getSharedPreferences(getPackageName(), ClientConnections.
             MODE_PRIVATE);
147      myPrefsEditor = myPrefs.edit();
148      String topicListJson = myPrefs.getString(clientHandle + ":" + "topicListJson",
             "");
149      Log.d("Con.Details:onGreate","handle ="+clientHandle+"/npreferences ="+
             topicListJson);
150      if (topicListJson!="") {
151        Type topicListType = new TypeToken<HashMap<String,String>>() {}.getType();
152        topicList = gson.fromJson(topicListJson,topicListType);
153      }
154
155
156    }
157    @Override
158    public void onPause() {
159      super.onPause();
160
161      String topicListJson = gson.toJson(topicList);
162      myPrefsEditor.putString(clientHandle + ":" + "topicListJson", topicListJson);
```

```
163       myPrefsEditor.commit();
164
165     }
166       @Override
167     protected void onDestroy() {
168       connection.removeChangeListener(null);
169       super.onDestroy();
170     }
171
172     /**
173      * @see android.app.Activity#onCreateOptionsMenu(android.view.Menu)
174      */
175     @Override
176     public boolean onCreateOptionsMenu(Menu menu) {
177       int menuID;
178       Integer button = null;
179       boolean connected = Connections.getInstance(this)
180           .getConnection(clientHandle).isConnected();
181
182       // Select the correct action bar menu to display based on the
183       // connectionStatus and which tab is selected
184       if (connected) {
185
186         switch (selected) {
187           case 0 : // Display
188             menuID = R.menu.activity_connection_details;
189             break;
190           case 1: // Insert Device
191             menuID = R.menu.activity_add_device;
192             button =R.id.addDevice;
193             break;
194           case 2 : // history
195             menuID = R.menu.activity_connection_details;
196             break;
197           default :
198             menuID = R.menu.activity_connection_details;
199             break;
200         }
201       }
202       else {
203         switch (selected) {
204           case 0: // Display
205             menuID = R.menu.activity_connection_details_disconnected;
206             break;
207           case 1: // Insert Device
208             menuID = R.menu.activity_add_device_disconnected;
209             button=R.id.addDevice;
210             break;
211           case 2: // history view
212             menuID = R.menu.activity_connection_details_disconnected;
213             break;
214           default :
215             menuID = R.menu.activity_connection_details_disconnected;
216             break;
217         }
218       }
219
220       // inflate the menu selected
221       getMenuInflater().inflate(menuID, menu);
222       Listener listener = new Listener(this, clientHandle);
223       // add listeners
224       menu.findItem(R.id.graphs).setOnMenuItemClickListener(listener);
225       if (button != null) {
226         // add listeners
227         menu.findItem(button).setOnMenuItemClickListener(listener);
228         if (!Connections.getInstance(this).getConnection(clientHandle)
229             .isConnected()) {
230           menu.findItem(button).setEnabled(false);
231         }
232       }
233       // add the listener to the disconnect or connect menu option
234       if (connected) {
235         menu.findItem(R.id.disconnect).setOnMenuItemClickListener(listener);
```

```
236       }
237       else {
238         menu.findItem(R.id.connectMenuOption).setOnMenuItemClickListener(
239             listener);
240       }
241
242       return true;
243     }
244
245     /**
246      * @see android.app.ActionBar.TabListener#onTabUnselected(android.app.ActionBar.
            Tab,
247      *      android.app.FragmentTransaction)
248      */
249     @Override
250     public void onTabUnselected(ActionBar.Tab tab,
251         FragmentTransaction fragmentTransaction) {
252       // Don't need to do anything when a tab is unselected
253     }
254
255     /**
256      * @see android.app.ActionBar.TabListener#onTabSelected(android.app.ActionBar.
            Tab,
257      *      android.app.FragmentTransaction)
258      */
259     @Override
260     public void onTabSelected(ActionBar.Tab tab,
261         FragmentTransaction fragmentTransaction) {
262       // When the given tab is selected, switch to the corresponding page in
263       // the ViewPager.
264       viewPager.setCurrentItem(tab.getPosition());
265       selected = tab.getPosition();
266       // invalidate the options menu so it can be updated
267       invalidateOptionsMenu();
268
269      ((LogFragment) sectionsPagerAdapter.getItem(2)).refresh();
270
271     }
272
273     /**
274      * @see android.app.ActionBar.TabListener#onTabReselected(android.app.ActionBar.
            Tab,
275      *      android.app.FragmentTransaction)
276      */
277     @Override
278     public void onTabReselected(ActionBar.Tab tab,
279         FragmentTransaction fragmentTransaction) {
280       // Don't need to do anything when the tab is reselected
281     }
282
283     /**
284      * Provides the Activity with the pages to display for each tab
285      *
286      */
287     public class SectionsPagerAdapter extends FragmentPagerAdapter {
288
289       // Stores the instances of the pages
290       private ArrayList<Fragment> fragments = null;
291
292       /**
293        * Only Constructor, requires a the activity's fragment managers
294        *
295        * @param fragmentManager
296        */
297       public SectionsPagerAdapter(FragmentManager fragmentManager) {
298         super(fragmentManager);
299         fragments = new ArrayList<Fragment>();
300
301         Bundle args = new Bundle();
302         args.putString("handle", getIntent().getStringExtra("handle"));
303         // add all the fragments for the display to the fragments list
304         Fragment  displaySensorsFragment = new DisplaySensorsFragment();
305         displaySensorsFragment.setArguments(args);
```

```
306         fragments.add(displaySensorsFragment);
307         Log.d("ConnDetails", "dSF ADDED");
308         fragments.add(new InsertDeviceFragment());
309         Fragment logFragment = new LogFragment();
310         logFragment.setArguments(args);
311         fragments.add(logFragment);
312         // CC end
313       }
314
315       /**
316        * @see android.support.v4.app.FragmentPagerAdapter#getItem(int)
317        */
318       @Override
319       public Fragment getItem(int position) {
320         return fragments.get(position);
321       }
322
323       /**
324        * @see android.support.v4.view.PagerAdapter#getCount()
325        */
326       @Override
327       public int getCount() {
328         return fragments.size();
329       }
330
331       /**
332        *
333        * @see FragmentPagerAdapter#getPageTitle(int)
334        */
335       @Override
336       public CharSequence getPageTitle(int position) {
337         switch (position) {
338           case 0 :
339             return "Display";
340           case 1:
341             return "Insert";
342           case 2 :
343             return "Log";
344         }
345         // return null if there is no title matching the position
346         return null;
347       }
348
349       @Override
350       public Object instantiateItem(ViewGroup container, int position) {
351         Fragment fragment = (Fragment) super.instantiateItem(container, position);
352         fragments.set(position,fragment);
353         return fragment;
354       }
355
356     }
357
358     /**
359      * <code>ChangeListener</code> updates the UI when the {@link Connection}
360      * object it is associated with updates
361      *
362      */
363     private class ChangeListener implements PropertyChangeListener {
364
365       /**
366        * @see java.beans.PropertyChangeListener#propertyChange(java.beans.
367            PropertyChangeEvent)
368        */
369       @Override
370       public void propertyChange(final PropertyChangeEvent event) {
371         // connection object has change refresh the UI
372
373         connectionDetails.runOnUiThread(new Runnable() {
374
375           @Override
376           public void run() {
377             connectionDetails.invalidateOptionsMenu();
```

```
            //Chrysostomos Christou: extra notification to change sensors UI
            if (event.getPropertyName().equals("topic")) {
                Log.d("ConnectionDetails", "propertyChange-topic");
                ((DisplaySensorsFragment) connectionDetails.sectionsPagerAdapter
                        .getItem(0)).refreshDeviceView((String) event.getOldValue(),
                             (String) event.getNewValue());
            }else{
              Log.d("ConnectionDetails", "propertyChange-history");
              ((LogFragment) connectionDetails.sectionsPagerAdapter
                      .getItem(2)).refresh();

            }
            if (event.getPropertyName().equals(ActivityConstants.
                 ConnectionStatusProperty)) {
              ((DisplaySensorsFragment) connectionDetails.sectionsPagerAdapter.
                  getItem(0)).listAdapter.notifyDataSetChanged();
            }

          }
        });

    }
}

}
```

B.4 Graphs.java

```
/*
 * Chrysostomos Christou, 2016, Thesis
 */
package com.chrysostomos.homemonitoring;


import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.FragmentTransaction;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.webkit.WebView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.ProgressBar;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttSecurityException;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Objects;

/**
 * Created by Chrysostomos on 09-May-16.
 */
public class Graphs extends MyBaseActivity  {

    private String clientHandle = null;
    WebView webView;
    ProgressBar progressBar ;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
45          setContentView(R.layout.activity_graphs);
46
47
48          clientHandle=getIntent().getStringExtra("clientHandle");
49          ArrayList<String> topicList = (ArrayList<String>) getIntent().
                getSerializableExtra("topicList");
50          topicList.add("Choose Device");
51
52          webView = (WebView) findViewById(R.id.webView);
53          progressBar = (ProgressBar) findViewById(R.id.progressBar);
54
55          Button loadBtn = (Button) findViewById(R.id.btnLoad);
56          Spinner devicesSpinner = (Spinner) findViewById(R.id.devicesSpinner);
57          TextView startDate = (TextView) findViewById(R.id.startDate);
58          TextView endDate = (TextView) findViewById(R.id.endDate);
59          loadBtn.setOnClickListener(listener);
60          startDate.setOnClickListener(listener);
61          endDate.setOnClickListener(listener);
62
63          HintAdapter<String> devicesAdapter = new HintAdapter<String>(this, android
                .R.layout.simple_spinner_item, topicList);
64          devicesSpinner.setAdapter(devicesAdapter);
65          devicesSpinner.setSelection(devicesAdapter.getCount());
66
67
68      }
69      @Override
70      protected void onResume(){
71          super.onResume();
72          String mqttTopic="Home/Commands/Graph";
73          String[] topics = new String[1];
74          topics[0] = mqttTopic;
75          int qos=2;
76          try {
77              Connections.getInstance(getApplicationContext()).getConnection(
                    clientHandle).getClient()
78                      .subscribe(mqttTopic, qos, null, new ActionListener(
                            getApplicationContext(), ActionListener.Action.SUBSCRIBE,
                            clientHandle, topics));
79          } catch (MqttSecurityException e) {
80              Log.e(this.getClass().getCanonicalName(), "Failed to subscribe to" +
                    mqttTopic + " the client with the handle " + clientHandle, e);
81          } catch (MqttException e) {
82              Log.e(this.getClass().getCanonicalName(), "Failed to subscribe to" +
                    mqttTopic + " the client with the handle " + clientHandle, e);
83          }
84      }
85
86      @Override
87      protected void onStop() {
88          super.onStop();
89          String mqttTopic="Home/Commands/Graph";
90          try {
91              Connections.getInstance(getApplicationContext()).getConnection(
                    clientHandle).getClient()
92                      .unsubscribe(mqttTopic,null,new ActionListener(
                            getApplicationContext(), ActionListener.Action.UNSUBSCRIBE
                            , clientHandle, mqttTopic));
93          } catch (MqttSecurityException e) {
94              Log.e(this.getClass().getCanonicalName(), "Failed to unsubscribe to" +
                     mqttTopic + " the client with the handle " + clientHandle, e);
95          } catch (MqttException e) {
96              Log.e(this.getClass().getCanonicalName(), "Failed to unsubscribe to" +
                     mqttTopic + " the client with the handle " + clientHandle, e);
97          }
98      }
99
100
101     public View.OnClickListener listener =new View.OnClickListener() {
102         @Override
103         public void onClick(View v) {
104
105             switch (v.getId()) {
```

```
106                 case R.id.btnLoad:
107                     String topic =null;
108                     if (((Spinner) findViewById(R.id.devicesSpinner)).
                            getSelectedItem()!=null)
109                         topic=((Spinner) findViewById(R.id.devicesSpinner)).
                                getSelectedItem().toString();
110
111                     String startDate =((TextView) findViewById(R.id.startDate)).
                            getText().toString();
112                     String endDate =((TextView) findViewById(R.id.endDate)).
                            getText().toString();
113                     if (topic==null||topic.equals("Choose Device")||startDate.
                            equals("")||endDate.equals(""))
114                     {
115                         (new Notify()).toast(Graphs.this,"Please Fill all Inputs",
                                Toast.LENGTH_SHORT);
116                     }else {
117
118                         StringBuilder tempMsg = new StringBuilder();
119                         tempMsg.append(topic + ",");
120                         tempMsg.append(startDate + ",");
121                         tempMsg.append(endDate);
122                         String msg = tempMsg.toString();
123                         String mqttTopic="Home/Commands/Graph";
124                         int qos = 2;
125                         boolean retained = false;
126                         String[] args = new String[2];
127                         args[0] = msg;
128                         args[1] = mqttTopic + ";qos:" + qos + ";retained:" +
                                retained;
129                         try {
130                             Connections.getInstance(getApplicationContext()).
                                    getConnection(clientHandle).getClient()
131                                     .publish(mqttTopic,msg.getBytes(), 2, retained
                                            , null, new ActionListener(
                                            getApplicationContext(), ActionListener.
                                            Action.PUBLISH, clientHandle, args));
132                         } catch (MqttSecurityException e) {
133                             Log.e(this.getClass().getCanonicalName(), "Failed to
                                    publish a messged from the client with the handle
                                    " + clientHandle, e);
134                         } catch (MqttException e) {
135                             Log.e(this.getClass().getCanonicalName(), "Failed to
                                    publish a messged from the client with the handle
                                    " + clientHandle, e);
136                         }
137                         (new Notify()).toast(Graphs.this, "Loading...", Toast.
                                LENGTH_SHORT);
138                         webView.setVisibility(View.INVISIBLE);
139                         progressBar.setVisibility(View.VISIBLE);
140
141
142                     }
143                 /*
144
145                  */   break;
146                 case R.id.startDate:
147                     FragmentTransaction ftStart = getFragmentManager().
                            beginTransaction();
148                     DialogFragment frStartDate = new DatePickerDialogFragment((
                            TextView) v);
149                     frStartDate.show(ftStart, "date_dialog");
150                     break;
151                 case R.id.endDate:
152                     FragmentTransaction ftEnd = getFragmentManager().
                            beginTransaction();
153                     DialogFragment frEndDate = new DatePickerDialogFragment((
                            TextView) v);
154                     frEndDate.show(ftEnd, "date_dialog");
155                     break;
156             }
157         }
158     };
```

```
159
160     public void refresh() {
161             String host =  Connections.getInstance(getApplicationContext()).
                    getConnection(clientHandle).getHostName();
162             webView.setVisibility(View.VISIBLE);
163             webView.getSettings().setJavaScriptEnabled(true);
164             webView.getSettings().setDomStorageEnabled(true);
165             webView.loadUrl("http://"+host + "/viewGraph/graph.html");
166             webView.getSettings().setBuiltInZoomControls(true);
167             progressBar.setVisibility(View.GONE);
168
169     }
170
171
172     public class HintAdapter<Objects> extends ArrayAdapter<Objects> {
173
174
175         public HintAdapter(Context context, int resource, List<Objects> objects) {
176             super(context, resource, objects);
177         }
178
179         @Override
180         public int getCount() {
181             // don't display last item. It is used as hint.
182             int count = super.getCount();
183             return count > 0 ? count - 1 : count;
184         }
185     }
186
187     public class DatePickerDialogFragment extends DialogFragment implements
            DatePickerDialog.OnDateSetListener {
188
189         private TextView mView;
190
191         public DatePickerDialogFragment() {
192             // nothing to see here, move along
193         }
194
195         public DatePickerDialogFragment(TextView view) {
196             mView = view;
197         }
198
199         public Dialog onCreateDialog(Bundle savedInstanceState) {
200             Calendar cal = Calendar.getInstance();
201
202             return new DatePickerDialog(getActivity(),
203                     this, cal.get(Calendar.YEAR),
204                     cal.get(Calendar.MONTH), cal.get(Calendar.DAY_OF_MONTH));
205         }
206
207         @Override
208         public void onDateSet(DatePicker view, int year, int monthOfYear, int
                dayOfMonth) {
209             Calendar cal = new GregorianCalendar(year, monthOfYear, dayOfMonth);
210             String formattedDate = new SimpleDateFormat("yyyy-MM-dd").format(cal.
                    getTime());
211             mView.setText(formattedDate);
212         }
213
214     }
215
216 }
```

B.5 DisplaySensorsFragment.java

```
1 /*
2  * Chrysostomos Christou, 2016, Thesis
3  */
4 package com.chrysostomos.homemonitoring;
5
6 import android.app.AlertDialog;
7 import android.content.Context;
```

```
8  import android.content.DialogInterface;
9  import android.content.SharedPreferences;
10 import android.os.Bundle;
11 import android.support.v4.app.Fragment;
12 import android.util.Log;
13 import android.view.LayoutInflater;
14 import android.view.View;
15 import android.view.ViewGroup;
16 import android.widget.AdapterView;
17 import android.widget.BaseExpandableListAdapter;
18 import android.widget.CompoundButton;
19 import android.widget.ExpandableListView;
20 import android.widget.Switch;
21 import android.widget.TextView;
22 import android.widget.Toast;
23
24 import com.google.gson.Gson;
25 import com.google.gson.GsonBuilder;
26 import com.google.gson.annotations.Expose;
27 import com.google.gson.reflect.TypeToken;
28
29 import org.eclipse.paho.client.mqttv3.MqttException;
30 import org.eclipse.paho.client.mqttv3.MqttSecurityException;
31
32 import java.lang.reflect.Type;
33 import java.text.SimpleDateFormat;
34 import java.util.ArrayList;
35 import java.util.Date;
36 import java.util.LinkedHashMap;
37
38 /**
39  * Fragment for the Example pane.
40  *
41  */
42 public class DisplaySensorsFragment extends Fragment  {
43
44     /** Client handle to a {@link Connection} object **/
45     String clientHandle = null;
46
47     Connection connection = null;
48     private View v;
49
50     private LinkedHashMap<String, RoomInfo> myRooms = new LinkedHashMap<String,
           RoomInfo>();
51     private ArrayList<RoomInfo> roomsList = new ArrayList<RoomInfo>();
52
53     public MyExpendableListAdapter listAdapter;
54     private ExpandableListView myListView ;
55
56     private SharedPreferences myPrefs;
57     private SharedPreferences.Editor myPrefsEditor;
58     private Gson gson = new GsonBuilder().excludeFieldsWithoutExposeAnnotation().
           create();
59
60     @Override
61     public void onCreate(Bundle savedInstanceState) {
62         super.onCreate(savedInstanceState);
63         // retain this fragment
64         setRetainInstance(true);
65         clientHandle = getArguments().getString("handle");
66
67         myPrefs = getActivity().getSharedPreferences(getActivity().getPackageName
               (), getActivity().getApplicationContext().MODE_PRIVATE);
68         myPrefsEditor = myPrefs.edit();
69         String myRoomsJson = myPrefs.getString(clientHandle + ":" + "myRoomsJson",
               "");
70         String roomListJson = myPrefs.getString(clientHandle + ":" + "roomListJson
               ", "");
71
72         if ((myRoomsJson != "") || (roomListJson != "")) {
73
74             Type myRoomsType = new TypeToken<LinkedHashMap<String, RoomInfo>>() {
75             }.getType();
```

```
76              Type roomListType = new TypeToken<ArrayList<RoomInfo>>() {
77              }.getType();
78              myRooms = gson.fromJson(myRoomsJson, myRoomsType);
79              roomsList = gson.fromJson(roomListJson, roomListType);
80
81          }
82
83      }
84
85      /**
86       *
87       * @return yyyy-MM-dd HH:mm:ss formate date as string
88       */
89      public static String getCurrentTimeStamp(){
90          try {
91
92              SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:
                    ss");
93              String currentTimeStamp = dateFormat.format(new Date()); // Find
                    todays date
94
95              return currentTimeStamp;
96          } catch (Exception e) {
97              e.printStackTrace();
98
99              return null;
100         }
101     }
102     public AdapterView.OnItemLongClickListener listener= new AdapterView.
            OnItemLongClickListener() {
103         @Override
104         public boolean onItemLongClick(AdapterView<?> adapterView, View view, int
                position, long id) {
105             if (!Connections.getInstance(getContext()).getConnection(clientHandle)
                    .isConnected()){
106                 (new Notify()).toast(getContext(), "Please Connect to delete
                        device", Toast.LENGTH_SHORT);
107                 return false;
108             }
109             if (ExpandableListView.getPackedPositionType(id) == ExpandableListView
                    .PACKED_POSITION_TYPE_CHILD) {
110                 final int groupPosition = ExpandableListView.
                        getPackedPositionGroup(id);
111                 final int childPosition = ExpandableListView.
                        getPackedPositionChild(id);
112                 final DeviceInfo deviceInfo = (DeviceInfo) listAdapter.getChild(
                        groupPosition, childPosition);
113                 final RoomInfo roomInfo = (RoomInfo) listAdapter.getGroup(
                        groupPosition);
114                 AlertDialog.Builder alert = new AlertDialog.Builder(getActivity())
                        ;
115                 alert.setTitle("Alert!!");
116                 alert.setMessage("Are you sure to delete this device?");
117                 alert.setPositiveButton("YES", new DialogInterface.OnClickListener
                        () {
118
119                     @Override
120                     public void onClick(DialogInterface dialog, int which) {
121                         ConnectionDetails connectionDetails = (ConnectionDetails)
                                getActivity();
122                         String mqttTopic="Home/"+roomInfo.getName()+"/"+deviceInfo
                                .getName();
123                         try {
124                             Connections.getInstance(getContext()).getConnection(
                                    clientHandle).getClient()
125                                     .unsubscribe(mqttTopic);
126                             connectionDetails.topicList.remove(mqttTopic);
127                             deleteDeviceView(roomInfo, deviceInfo);
128                         } catch (MqttSecurityException e) {
129                             Log.e(this.getClass().getCanonicalName(), "Failed to
                                    unsubscribe to" + mqttTopic + " the client with
                                    the handle " + clientHandle, e);
130                         } catch (MqttException e) {
```

```
131                             Log.e(this.getClass().getCanonicalName(), "Failed to
                                    unsubscribe to" + mqttTopic + " the client with
                                    the handle " + clientHandle, e);
132                         }
133
134                         dialog.dismiss();
135
136                     }
137                 });
138                 alert.setNegativeButton("NO", new DialogInterface.OnClickListener
                        () {
139
140                     @Override
141                     public void onClick(DialogInterface dialog, int which) {
142                         dialog.dismiss();
143                     }
144                 });
145
146                 alert.show();
147
148                 return true;
149             }
150
151             return false;
152         }
153     };
154
155     @Override
156     public void onPause() {
157         super.onPause();
158
159         String myRoomsJson = gson.toJson(myRooms);
160         String roomListJson = gson.toJson(roomsList);
161         myPrefsEditor.putString(clientHandle + ":" + "myRoomsJson", myRoomsJson);
162         myPrefsEditor.putString(clientHandle + ":" + "roomListJson", roomListJson)
                ;
163         myPrefsEditor.commit();
164
165
166     }
167
168     /**
169     * @see Fragment#onCreateView(LayoutInflater, ViewGroup, Bundle)
170     */
171     @Override
172     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
            savedInstanceState) {
173
174
175         v = inflater.inflate(R.layout.activity_display_sensors, null);
176
177         //get reference to the ExpandableListView
178         myListView = (ExpandableListView) v.findViewById(R.id.list);
179         //create the adapter by passing your ArrayList data
180         listAdapter = new MyExpendableListAdapter(getActivity(), roomsList);
181         //attach the adapter to the list
182         myListView.setAdapter(listAdapter);
183         myListView.setOnItemLongClickListener(listener);
184         expandAll();
185
186       return v;
187   }
188     public void setData(View myView) {
189         this.v = myView;
190     }
191
192     public View getData() {
193         return v;
194     }
195
196     public void displayDevice(String room, String device, String type) {
197
198         //check the hash map if the group already exists
```

```
199         RoomInfo roomInfoMap = myRooms.get(room);
200          //add the group if doesn't exists
201          if(roomInfoMap == null){
202              roomInfoMap = new RoomInfo();
203              roomInfoMap.setName(room);
204              roomInfoMap.setPosition(roomsList.size());
205              myRooms.put(room, roomInfoMap);
206              roomsList.add(roomInfoMap);
207          }
208
209         //create a new child and add that to the group
210         DeviceInfo deviceInfo = new DeviceInfo();
211          deviceInfo.setType(type);
212         deviceInfo.setName(device);
213
214         deviceInfo.setValue(getString(R.string.noValue));
215         roomInfoMap.setDevice(device, deviceInfo);
216         roomsList.set(roomInfoMap.getPosition(), roomInfoMap);
217         listAdapter.notifyDataSetChanged();
218
219
220
221          //collapse all groups
222          collapseAll();
223          //expand the group where item was just added
224          myListView.expandGroup(roomInfoMap.getPosition());
225          //set the current group to be selected so that it becomes visible
226          myListView.setSelection(roomInfoMap.getPosition());
227
228     }
229
230     public void deleteDeviceView(RoomInfo roomInfo,DeviceInfo deviceInfo) {
231
232
233          roomInfo.delDevice(deviceInfo.getName());
234          if (roomInfo.getSize()==0) {
235              roomsList.remove(roomInfo);
236              myRooms.remove(roomInfo.getName());
237              Log.d("DFS: delDeviceView", "room size = 0");
238          }else{
239              roomsList.set(roomInfo.getPosition(), roomInfo);
240              Log.d("DFS: delDeviceView", "room size >0");
241          }
242          listAdapter.notifyDataSetChanged();
243     }
244
245     public void refreshDeviceView(String topic,String value) {
246          String[] paths=topic.split("/");
247          String roomName=paths[1];
248          String deviceName =paths[2];
249          RoomInfo roomInfo = myRooms.get(roomName);
250          roomInfo.getDevice(deviceName).setValue(value);
251          roomsList.set(roomInfo.getPosition(),roomInfo);
252          listAdapter.notifyDataSetChanged();
253          //collapse all groups
254          collapseAll();
255          //expand the group where item was just added
256          myListView.expandGroup(roomsList.indexOf(roomInfo));
257          //set the current group to be selected so that it becomes visible
258          myListView.setSelection(roomInfo.getPosition());
259     }
260
261     //method to expand all groups
262     private void expandAll() {
263          int count = listAdapter.getGroupCount();
264          for (int i = 0; i < count; i++){
265              myListView.expandGroup(i);
266          }
267     }
268
269     //method to collapse all groups
270     private void collapseAll() {
271          int count = listAdapter.getGroupCount();
```

```
272         for (int i = 0; i < count; i++){
273             myListView.collapseGroup(i);
274         }
275     }
276
277     private class DeviceInfo {
278         @Expose
279         private String type = "";
280         @Expose
281         private String value = "";
282         @Expose
283         private String name = "";
284
285         public String getType() {
286             return type;
287         }
288         public void setType(String type) {
289             this.type = type;
290         }
291         public String getValue() {
292             return value;
293         }
294         public void setValue(String value) {
295             this.value = value;
296         }
297         public String getName() {
298             return name;
299         }
300         public void setName(String name) {
301             this.name = name;
302         }
303     }
304
305     private class RoomInfo {
306
307         @Expose
308         private String name;
309         @Expose
310         private Integer position;
311         @Expose
312         public LinkedHashMap<String, DeviceInfo> myDevices = new LinkedHashMap<
                String, DeviceInfo>();
313
314         public String getName() {
315             return name;
316         }
317         public void setName(String name) {
318             this.name = name;
319         }
320         public Integer getPosition(){
321             return position;
322         }
323         public void setPosition(Integer position){
324             this.position=position;
325         }
326         public DeviceInfo getDevice(String device) {
327             return myDevices.get(device);
328         }
329         public void setDevice(String device,DeviceInfo deviceInfo) {
330             myDevices.put(device,deviceInfo);
331         }
332         public void delDevice(String device){myDevices.remove(device);}
333         public int getSize(){return  myDevices.size();}
334
335     }
336
337     public class MyExpendableListAdapter extends BaseExpandableListAdapter {
338
339         private Context context;
340         private ArrayList<RoomInfo> roomList;
341
342         public MyExpendableListAdapter(Context context, ArrayList<RoomInfo>
                roomList) {
```

```
343             this.context = context;
344             this.roomList = roomList;
345         }
346 
347         @Override
348         public int getChildrenCount(int groupPosition) {
349             return roomList.get(groupPosition).myDevices.size();
350 
351         }
352 
353         @Override
354         public RoomInfo getGroup(int groupPosition) {
355             return roomList.get(groupPosition);
356         }
357 
358         @Override
359         public int getGroupCount() {
360             return roomList.size();
361         }
362 
363         @Override
364         public long getGroupId(int groupPosition) {
365             return groupPosition;
366         }
367 
368         @Override
369         public boolean hasStableIds() {
370             return true;
371         }
372 
373         @Override
374         public boolean isChildSelectable(int groupPosition, int childPosition) {
375             return true;
376         }
377 
378         @Override
379         public Object getChild(int groupPosition, int childPosition) {
380            return this.getGroup(groupPosition).myDevices.values().toArray()[
                   childPosition];
381         }
382 
383         @Override
384         public long getChildId(int groupPosition, int childPosition) {
385             return childPosition;
386         }
387 
388         @Override
389         public View getGroupView(int groupPosition, boolean isLastChild, View view
                ,
390                                  ViewGroup parent) {
391 
392             RoomInfo roomInfo = (RoomInfo) getGroup(groupPosition);
393             if (view == null) {
394                 LayoutInflater inf = (LayoutInflater) context.getSystemService(
                        Context.LAYOUT_INFLATER_SERVICE);
395                 view = inf.inflate(R.layout.listrow_group, null);
396             }
397 
398             TextView name = (TextView) view.findViewById(R.id.txtRoom);
399             name.setText(roomInfo.getName());
400 
401             return view;
402         }
403 
404         @Override
405         public View getChildView(int groupPosition, int childPosition, boolean
                isLastChild, View view, ViewGroup parent) {
406 
407             final RoomInfo roomInfo =  getGroup(groupPosition);
408             final DeviceInfo deviceInfo = (DeviceInfo) getChild(groupPosition,
                    childPosition);
409 
410             LayoutInflater infalInflater = (LayoutInflater) context.
```

```
                getSystemService(Context.LAYOUT_INFLATER_SERVICE);
411             if (view==null){
412                 view = infalInflater.inflate(R.layout.listrow_input, null);
413             }
414             TextView name = (TextView) view.findViewById(R.id.txtName);
415             TextView value = (TextView) view.findViewById(R.id.txtValue);
416             Switch switchBtn = (Switch) view.findViewById(R.id.switch1);
417
418             switch (deviceInfo.getType()){
419                 case "Two State - Input":
420                     name.setText(deviceInfo.getName());
421                     value.setText(deviceInfo.getValue());
422                     value.setVisibility(View.VISIBLE);
423                     switchBtn.setVisibility(View.GONE);
424                     break;
425                 case "Multi State - Input":
426                     name.setText(deviceInfo.getName());
427                     value.setText(deviceInfo.getValue());
428                     value.setVisibility(View.VISIBLE);
429                     switchBtn.setVisibility(View.GONE);
430                     break;
431                 case "Two State - Output":
432                     name.setText(deviceInfo.getName());
433                     switchBtn.setVisibility(View.VISIBLE);
434                     String[] components = deviceInfo.getValue().split("@");
435                     switchBtn.setOnCheckedChangeListener(null);
436                     switchBtn.setChecked(components[0].equals("ON"));
437                     if (components.length>1)
438                         value.setText("@" + components[1]);
439                     else
440                         value.setText("@"+components[0]);
441                     if (Connections.getInstance(getContext()).getConnection(
                            clientHandle).isConnected())
442                         switchBtn.setEnabled(true);
443                     else
444                         switchBtn.setEnabled(false);
445                     switchBtn.setOnCheckedChangeListener(new CompoundButton.
                            OnCheckedChangeListener() {
446                         @Override
447                         public void onCheckedChanged(CompoundButton buttonView,
                                boolean isChecked) {
448                             Log.d("listener","checkedchange");
449                             // TODO Auto-generated method stub
450                             String msg;
451                             if (buttonView.isChecked()){
452                                 msg="ON@"+getCurrentTimeStamp();
453                             }
454                             else{
455                                 msg="OFF@"+getCurrentTimeStamp();
456                             }
457
458                             String mqttTopic="Home/"+roomInfo.getName()+"/"+
                                    deviceInfo.getName();
459                             int qos = 2;
460                             boolean retained = true;
461                             String[] args = new String[2];
462                             args[0] = msg;
463                             args[1] = mqttTopic + ";qos:" + qos + ";retained:" +
                                    retained;
464                             try {
465                                 Connections.getInstance(getActivity().
                                        getApplicationContext()).getConnection(
                                        clientHandle).getClient()
466                                         .publish(mqttTopic,msg.getBytes(), 2,
                                                retained, null, new ActionListener(
                                                getActivity().getApplicationContext(),
                                                 ActionListener.Action.PUBLISH,
                                                clientHandle, args));
467                             } catch (MqttSecurityException e) {
468                                 Log.e(this.getClass().getCanonicalName(), "Failed
                                        to publish a messged from the client with the
                                        handle " + clientHandle, e);
469                             } catch (MqttException e) {
```

```
470                                     Log.e(this.getClass().getCanonicalName(), "Failed
                                            to publish a messged from the client with the
                                            handle " + clientHandle, e);
471                                 }
472                             }
473                         });
474                         break;
475                 }
476
477
478                 return view;
479             }
480
481
482         }
483
484 }
```

B.6 InsertDeviceFragment.java

```
1  /*
2   * Chrysostomos Christou, 2016, Thesis
3   */
4  package com.chrysostomos.homemonitoring;
5
6  import android.os.Bundle;
7  import android.support.v4.app.Fragment;
8  import android.view.LayoutInflater;
9  import android.view.View;
10 import android.view.ViewGroup;
11
12 /**
13  * Created by Chrysostomos on 06-Apr-16.
14  */
15 public class InsertDeviceFragment  extends Fragment {
16
17     View v;
18
19     /**
20      * @see Fragment#onCreateView(LayoutInflater, ViewGroup, Bundle)
21      */
22     @Override
23     public View onCreateView(LayoutInflater inflater, ViewGroup container,
24                              Bundle savedInstanceState) {
25       //  super.onCreateView(inflater,container,savedInstanceState);
26
27         v = inflater.inflate(R.layout.activity_insert_device, null);
28         return v;
29     }
30
31 }
```

B.7 LogFragment.java

```
1  /*
2   * Chrysostomos Christou, 2016, Thesis
3   */
4  package com.chrysostomos.homemonitoring;
5
6  import android.os.Bundle;
7  import android.support.v4.app.ListFragment;
8  import android.text.Spanned;
9  import android.util.Log;
10 import android.widget.ArrayAdapter;
11
12 /**
13  * This fragment displays the history information for a client
14  *
15  */
16 public class LogFragment extends ListFragment {
17
```

```
18    /** Client handle to a {@link Connection} object **/
19    String clientHandle = null;
20    /** {@link ArrayAdapter} to display the formatted text **/
21    ArrayAdapter<Spanned> arrayAdapter = null;
22
23    /**
24     * @see ListFragment#onCreate(Bundle)
25     */
26    @Override
27    public void onCreate(Bundle savedInstanceState) {
28      super.onCreate(savedInstanceState);
29
30      //Pull history information out of bundle
31
32      clientHandle = getArguments().getString("handle");
33      Connection connection = Connections.getInstance(getActivity()).getConnection(
            clientHandle);
34
35      Spanned[] history = connection.history();
36
37      //Initialise the arrayAdapter, view and add data
38      arrayAdapter = new ArrayAdapter<Spanned>(getActivity(), R.layout.
            list_view_text_view);
39
40      arrayAdapter.addAll(history);
41      setListAdapter(arrayAdapter);
42
43    }
44
45    /**
46     * Updates the data displayed to match the current history
47     */
48    public void refresh() {
49      if (arrayAdapter != null) {
50        arrayAdapter.clear();
51        arrayAdapter.addAll(Connections.getInstance(getActivity()).getConnection(
              clientHandle).history());
52        arrayAdapter.notifyDataSetChanged();
53      }
54
55    }
56
57  }
```

B.8 Listener.java

```
1  //Original work
2  /*******************************************************************************
3   * Copyright (c) 1999, 2014 IBM Corp.
4   *
5   * All rights reserved. This program and the accompanying materials
6   * are made available under the terms of the Eclipse Public License v1.0
7   * and Eclipse Distribution License v1.0 which accompany this distribution.
8   *
9   * The Eclipse Public License is available at
10  *    http://www.eclipse.org/legal/epl-v10.html
11  * and the Eclipse Distribution License is available at
12  *   http://www.eclipse.org/org/documents/edl-v10.php.
13  */
14 //Modified work
15 /*
16  * Chrysostomos Christou, 2016, Thesis
17  */
18 package com.chrysostomos.homemonitoring;
19
20 import android.content.Context;
21 import android.content.Intent;
22 import android.content.SharedPreferences;
23 import android.preference.PreferenceManager;
24 import android.util.Log;
25 import android.view.MenuItem;
26 import android.view.MenuItem.OnMenuItemClickListener;
```

```
27 import android.widget.Spinner;
28 import android.widget.Toast;
29
30 import org.eclipse.paho.android.service.MqttAndroidClient;
31 import com.chrysostomos.homemonitoring.ActionListener.Action;
32 import com.chrysostomos.homemonitoring.Connection.ConnectionStatus;
33
34 import org.eclipse.paho.client.mqttv3.MqttException;
35 import org.eclipse.paho.client.mqttv3.MqttSecurityException;
36
37 import java.io.FileInputStream;
38 import java.io.FileNotFoundException;
39 import java.util.ArrayList;
40
41 /**
42  * Deals with actions performed in the {@link ClientConnections} activity
43  * and the {@link ConnectionDetails} activity and associated fragments
44  *
45  */
46 public class Listener implements OnMenuItemClickListener {
47
48   /**
49    * The handle to a {@link Connection} object which contains the {@link
         MqttAndroidClient} associated with this object
50    **/
51   private String clientHandle = null;
52
53   /**
54    * {@link ConnectionDetails} reference used to perform some actions
55    **/
56   private ConnectionDetails connectionDetails = null;
57   /**
58    * {@link ClientConnections} reference used to perform some actions
59    **/
60   private ClientConnections clientConnections = null;
61   /**
62    * {@link Context} used to load and format strings
63    **/
64   private Context context = null;
65
66   /**
67    * Constructs a listener object for use with {@link ConnectionDetails} activity
         and
68    * associated fragments.
69    *
70    * @param connectionDetails The instance of {@link ConnectionDetails}
71    * @param clientHandle      The handle to the client that the actions are to be
         performed on
72    */
73   public Listener(ConnectionDetails connectionDetails, String clientHandle) {
74     this.connectionDetails = connectionDetails;
75     this.clientHandle = clientHandle;
76     context = connectionDetails;
77
78   }
79
80   /**
81    * Constructs a listener object for use with {@link ClientConnections} activity.
82    *
83    * @param clientConnections The instance of {@link ClientConnections}
84    */
85   public Listener(ClientConnections clientConnections) {
86     this.clientConnections = clientConnections;
87     context = clientConnections;
88   }
89
90   /**
91    * Perform the needed action required based on the button that
92    * the user has clicked.
93    *
94    * @param item The menu item that was clicked
95    * @return If there is anymore processing to be done
96    */
```

```
97    @Override
98    public boolean onMenuItemClick(MenuItem item) {
99
100     int id = item.getItemId();
101
102     switch (id) {
103       case R.id.newConnection:
104         createAndConnect();
105         break;
106       case R.id.disconnect:
107         disconnect();
108         break;
109       case R.id.connectMenuOption:
110         reconnect();
111         break;
112       case R.id.addDevice:
113         addDevice();
114         break;
115       case R.id.graphs:
116         graphs();
117     }
118
119     return false;
120   }
121
122   /**
123    * Reconnect the selected client
124    */
125   private void reconnect() {
126
127     Connections.getInstance(context).getConnection(clientHandle).
          changeConnectionStatus(ConnectionStatus.CONNECTING);
128
129     Connection c = Connections.getInstance(context).getConnection(clientHandle);
130
131     try {
132       if (c.isSSL()==1){
133         SharedPreferences myPrefs = PreferenceManager.getDefaultSharedPreferences(
              context);
134         String ssl_key = myPrefs.getString("ssl_key", "");
135         String ssl_pass = myPrefs.getString("ssl_pass", "");
136         Log.d("Listener:reconnect", "SSL_key:  : " + ssl_key);
137         Log.d("Listener:reconnect", "SSL_pass:  : " + ssl_pass);
138         if (ssl_key!="") {
139           if (c.getConnectionOptions().getSocketFactory() == null) {
140             Log.d("Listener", "reconnect: null socket");
141             FileInputStream key = new FileInputStream(ssl_key);
142             c.getConnectionOptions().setSocketFactory(c.getClient().
                  getSSLSocketFactory(key, ssl_pass));
143           } else Log.d("Listener", "reconnect: full socket");
144         }
145       }
146       c.getClient().connect(c.getConnectionOptions(), null, new ActionListener(
            context, Action.CONNECT, clientHandle, null));
147     } catch (MqttSecurityException e) {
148       Log.e(this.getClass().getCanonicalName(), "Failed to reconnect the client
            with the handle " + clientHandle, e);
149       c.addAction("Client failed to connect");
150     } catch (MqttException e) {
151       Log.e(this.getClass().getCanonicalName(), "Failed to reconnect the client
            with the handle " + clientHandle, e);
152       c.addAction("Client failed to connect");
153     } catch (FileNotFoundException e) {
154       Log.e(this.getClass().getCanonicalName(),
155               "MqttException Occured: SSL Key file not found", e);
156     }
157
158   }
159
160   /**
161    * Disconnect the client
162    */
163   private void disconnect() {
```

```
164
165     Connection c = Connections.getInstance(context).getConnection(clientHandle);
166
167     //if the client is not connected, process the disconnect
168     if (!c.isConnected()) {
169       return;
170     }
171
172     try {
173       c.getClient().disconnect(null, new ActionListener(context, Action.DISCONNECT
              , clientHandle, null));
174       c.changeConnectionStatus(ConnectionStatus.DISCONNECTING);
175     } catch (MqttException e) {
176       Log.e(this.getClass().getCanonicalName(), "Failed to disconnect the client
              with the handle " + clientHandle, e);
177       c.addAction("Client failed to disconnect");
178     }
179
180   }
181
182   /**
183    * Create a new client and connect
184    */
185   private void createAndConnect() {
186     Intent createConnection;
187
188     //start a new activity to gather information for a new connection
189     createConnection = new Intent();
190     createConnection.setClassName(
191             clientConnections.getApplicationContext(),
192             "com.chrysostomos.homemonitoring.NewConnection");
193
194     clientConnections.startActivityForResult(createConnection,
195             ActivityConstants.connect);
196   }
197
198
199   /**
200    * Subscribe to the selected device and add it to the display list
201    */
202   private void addDevice() {
203     String location = ((Spinner) connectionDetails.findViewById(R.id.
          locationSpinner)).getSelectedItem().toString();
204     String room = ((Spinner) connectionDetails.findViewById(R.id.roomSpinner)).
          getSelectedItem().toString();
205     String device = ((Spinner) connectionDetails.findViewById(R.id.deviceSpinner))
          .getSelectedItem().toString();
206     String type = ((Spinner) connectionDetails.findViewById(R.id.deviceTypeSpinner
          )).getSelectedItem().toString();
207
208
209     StringBuilder topic = new StringBuilder();
210     topic.append(location + "/");
211     topic.append(room + "/");
212     topic.append(device);
213     String[] topics = new String[1];
214     topics[0] = topic.toString();
215
216     if (connectionDetails.topicList.containsKey(topics[0])) {
217       (new Notify()).toast(connectionDetails, "Device Already Exist", Toast.
              LENGTH_SHORT);
218       return;
219     }
220
221
222     try {
223         int qos = 2;
224         Connections.getInstance(context).getConnection(clientHandle).getClient()
225                 .subscribe(topic.toString(), qos, null, new ActionListener(context
                        , Action.SUBSCRIBE, clientHandle, topics));
226       ((DisplaySensorsFragment) connectionDetails.sectionsPagerAdapter
227               .getItem(0)).displayDevice(room, device, type);
228       connectionDetails.topicList.put(topics[0], type);
```

```
229     } catch (MqttSecurityException e) {
230       Log.e(this.getClass().getCanonicalName(), "Failed to subscribe to" + topic +
              " the client with the handle " + clientHandle, e);
231     } catch (MqttException e) {
232       Log.e(this.getClass().getCanonicalName(), "Failed to subscribe to" + topic +
              " the client with the handle " + clientHandle, e);
233     }
234   }
235
236
237   public void graphs() {
238     Intent intent = new Intent(context, Graphs.class);
239     intent.putExtra("topicList",new ArrayList<String>(connectionDetails.topicList.
            keySet()));
240     intent.putExtra("clientHandle",clientHandle);
241     connectionDetails.startActivity(intent);
242   }
243 }
```

B.9 MqttCallbackHandler.java

```
1  //Original work
2  /*******************************************************************************
3   * Copyright (c) 1999, 2014 IBM Corp.
4   *
5   * All rights reserved. This program and the accompanying materials
6   * are made available under the terms of the Eclipse Public License v1.0
7   * and Eclipse Distribution License v1.0 which accompany this distribution.
8   *
9   * The Eclipse Public License is available at
10  *    http://www.eclipse.org/legal/epl-v10.html
11  * and the Eclipse Distribution License is available at
12  *   http://www.eclipse.org/org/documents/edl-v10.php.
13  */
14 //Modified work
15 /*
16  * Chrysostomos Christou, 2016, Thesis
17  */
18 package com.chrysostomos.homemonitoring;
19
20 import android.app.Activity;
21 import android.app.ActivityManager;
22 import android.content.ComponentName;
23 import android.content.Context;
24 import android.content.Intent;
25 import android.util.Log;
26
27 import com.chrysostomos.homemonitoring.Connection.ConnectionStatus;
28 import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
29 import org.eclipse.paho.client.mqttv3.MqttCallback;
30 import org.eclipse.paho.client.mqttv3.MqttMessage;
31
32 /**
33  * Handles call backs from the MQTT Client
34  *
35  */
36 public class MqttCallbackHandler implements MqttCallback {
37
38   /** {@link Context} for the application used to format and import external
         strings**/
39   private Context context;
40   /** Client handle to reference the connection that this handler is attached to*
         */
41   private String clientHandle;
42
43   /**
44    * Creates an <code>MqttCallbackHandler</code> object
45    * @param context The application's context
46    * @param clientHandle The handle to a {@link Connection} object
47    */
48   public MqttCallbackHandler(Context context, String clientHandle)
49   {
```

```
50      this.context = context;
51      this.clientHandle = clientHandle;
52    }
53
54    /**
55     * @see org.eclipse.paho.client.mqttv3.MqttCallback#connectionLost(java.lang.
          Throwable)
56     */
57    @Override
58    public void connectionLost(Throwable cause) {
59  //    cause.printStackTrace();
60      if (cause != null) {
61        Connection c = Connections.getInstance(context).getConnection(clientHandle);
62        c.addAction("Connection Lost");
63        c.changeConnectionStatus(ConnectionStatus.DISCONNECTED);
64
65        //format string to use a notification text
66        Object[] args = new Object[2];
67        args[0] = c.getId();
68        args[1] = c.getHostName();
69
70        String message = context.getString(R.string.connection_lost, args);
71
72        //build intent
73        Intent intent = new Intent();
74        intent.setClassName(context, "com.chrysostomos.homemonitoring.
              ConnectionDetails");
75        intent.putExtra("handle", clientHandle);
76
77        //notify the user
78        Notify.notifcation(context, message, intent, R.string.
              notifyTitle_connectionLost);
79      }
80    }
81
82    /**
83     * @see org.eclipse.paho.client.mqttv3.MqttCallback#messageArrived(java.lang.
          String, org.eclipse.paho.client.mqttv3.MqttMessage)
84     */
85    @Override
86    public void messageArrived(String topic, MqttMessage message) throws Exception {
87
88
89      //Get connection object associated with this object
90      Connection c = Connections.getInstance(context).getConnection(clientHandle);
91
92      //create arguments to format message arrived notifcation string
93      String[] args = new String[2];
94      args[0] = new String(message.getPayload());
95      args[1] = topic+";qos:"+message.getQos()+";retained:"+message.isRetained();
96      Log.d("MqttCallbackHandler","messageArrived "+ topic + " : "+args[0]);
97      //get the string from strings.xml and format
98      String messageString = context.getString(R.string.messageRecieved, (Object[])
            args);
99
100     //create intent to start activity
101     Intent intent = new Intent();
102     intent.setClassName(context, "com.chrysostomos.homemonitoring.
            ConnectionDetails");
103     intent.putExtra("handle", clientHandle);
104
105     //format string args
106     Object[] notifyArgs = new String[3];
107     notifyArgs[0] = c.getId();
108     notifyArgs[1] = new String(message.getPayload());
109     notifyArgs[2] = topic;
110
111
112     if (topic.equals("Home/Commands/Graph")&args[0].equals("Ready")){
113
114       Activity currentActivity = ((MyApp)context.getApplicationContext()).
              getCurrentActivity();
115       Graphs graphs = (Graphs) currentActivity;
```

```
116             graphs.refresh();
117
118         }else {
119             //update client Log
120             c.addAction(messageString);
121             // extra action to change sensors UI
122             c.addAction((String) notifyArgs[2], (String) notifyArgs[1]);
123
124            // notify the user
125             Notify.notifcation(context, context.getString(R.string.notification,
                    notifyArgs), intent, R.string.notifyTitle);
126
127         }
128     }
129
130     /**
131      * @see org.eclipse.paho.client.mqttv3.MqttCallback#deliveryComplete(org.eclipse
                .paho.client.mqttv3.IMqttDeliveryToken)
132      */
133     @Override
134     public void deliveryComplete(IMqttDeliveryToken token) {
135         // Do nothing
136     }
137
138 }
```