Ατομική Διπλωματική Εργασία

ΔΙΑΧΕΙΡΙΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΑΙΣΘΗΤΗΡΩΝ ΜΕΣΩ ΠΛΑΤΦΟΡΜΑΣ ΜΙΚΡΟΕΛΕΓΚΤΗ RASPBERRY PI ΚΑΙ ΕΦΑΡΜΟΓΗΣ ΣΕ ANDROID

Αθηνά Παφίτου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση πληροφοριών αισθητήρων μέσω πλατφόρμας μικροελεγκτή και εφαρμογής σε Android

Αθηνά Παφίτου

Επιβλέπουσα Καθηγήτρια

Γεωργία Καπιτσάκη

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους συμφοιτητές μου Στέλλα Κωνσταντίνου και Κυριάκο Γεωργίου για την βοήθεια που προσέφεραν στην διεκπεραίωση της ΑΔΕ.

Περίληψη

Τα δίκτυα αισθητήρων (sensor networks) μας δίνουν την δυνατότητα συλλογής δεδομένων από το περιβάλλον και βοηθούν στην ανάπτυξη εφαρμογών με επίγνωση συγκειμένου (context-aware εφαρμογών). Παράλληλα, η χρήση πλατφόρμων με μικροελεγκτές και είσοδο/ έξοδο για σύνδεση με εξωτερικά κυκλώματα αισθητήρων έχει διαδοθεί εκτενώς. Ο συνδυασμός πλατφόρμων μικροελεγκτών όπως το Arduino και το Raspberry Pi με κυκλώματα αισθητήρων που υπάρχουν στο περιβάλλον ή σε εξωτερικές συσκευές βοηθά σημαντικά στην συλλογή πληροφοριών περιβάλλοντος και στη δημιουργία εφαρμογών με επίγνωση συγκειμένου.

Σκοπός αυτής της διπλωματικής εργασίας αποτελεί η χρήση τέτοιων πλατφόρμων μικροελεγκτών και ο συνδυασμός της χρήσης τους με μία εφαρμογή κινητών συσκευών Android.

Συγκεκριμένα, μέσω της εφαρμογής παρέχεται η δυνατότητα ανάκτησης των δεδομένων των αισθητήρων που είναι ενωμένοι σε πλατφόρμες μικροελεγκτών μέσω υπηρεσιών διαδικτύου. Αυτή η ανάκτηση δεδομένων γίνεται μέσω κάποιων επιλογών του χρήστη όπως για παράδειγμα η τοποθεσία και ο τύπος αναζήτησης που επιλέγει. Επιπλέον, η εφαρμογή επιτρέπει στο χρήστη τον έλεγχο διάφορων παραμέτρων καταγραφής των δεδομένων από τους αισθητήρες όπως η συχνότητα συλλογής τους και η ενεργοποίηση/ απενεργοποίηση αισθητήρων. Πέραν αυτού, μέσω της εφαρμογής δίδεται η δυνατότητα λήψης ιστορικών μετρήσεων αισθητήρων και προβολή των μετρήσεων με γραφικές παραστάσεις.

Επίσης, η εφαρμογή παρέχει την ευελιξία προσθήκης νέων χρηστών στο σύστημα καθώς επίσης και την προσθήκη αισθητήρων ή πλατφόρμων μικροελεγκτών για καταγραφή των δεδομένων τους.

Τέλος, μέσω της εφαρμογής παρέχεται σύνδεση με την εφαρμογή Anyplace. Κάνοντας χρήση του Anyplace ο χρήστης έχει τη δυνατότητα να προσθέσει στην εφαρμογή διάφορες γεωγραφικές τοποθεσίες που έχουν ιδιαίτερο ενδιαφέρον (Points Of Interest) έτσι ώστε να έχει εύκολη πρόσβαση στην τοποθεσία τους.

Η συγκεκριμένη εφαρμογή μπορεί να συνδυαστεί με διάφορες εφαρμογές για την αποδοτική χρήση των δεδομένων που συλλέγονται από τις πλατφόρμες μικροελεγκτών σε εφαρμογές με επίγνωση συγκειμένου.

Περιεχόμενα

Εικόνες

1 Εισαγωγή

1.1 Γενική εισαγωγή

Η διαχείριση αισθητήρων διάσπαρτων στο περιβάλλον μπορεί να είναι μια πολύτιμη διαδικασία για μια πληθώρα εφαρμογών σε πεδία όπως γεωργία, έξυπνα σπίτια, ιατρική τεχνολογία και παροχή ιατρικής βοήθειας, όπου η επίγνωση συγκειμένου και η αναγνώριση του περιβάλλοντος είναι αναγκαία. Η αυξανόμενη διασημότητα του Internet of Things (IoT) έχει αλλάξει τον τρόπο που βλέπουμε την επίγνωση συγκειμένου καθώς υπόσχεται τη δημιουργία ενός κόσμου όπου όλα τα αντικείμενα γύρω μας είναι ενωμένα στο διαδίκτυο και επικοινωνούν μεταξύ τους με ελάχιστη ανθρώπινη παρέμβαση.

Ταυτόχρονα, η ευρεία εξάπλωση πλατφόρμων με μικροελεγκτές με υπερισχύουσες επιλογές το Arduino και το Raspberry Pi επηρεάζουν τον τρόπο υλοποίησης τέτοιων εφαρμογών. Αυτές οι πλατφόρμες επιτρέπουν τη διασύνδεση μιας ποικιλίας αισθητήρων διαφόρων μεγεθών, λειτουργικότητας και ακρίβειας προσφέροντας μια σφαιρική όψη του περιβάλλοντος γύρω τους.

Επιπρόσθετα, για να υπάρχει η δυνατότητα ενιαίας χρήσης αυτής της πληροφορίας σε διαφορετικές περιστάσεις, είναι απαραίτητη η ανάπτυξη ευφυών μηχανισμών που μπορούν να λειτουργούν αποτελεσματικά κατά την ενσωμάτωση τους σε συστήματα που μπορούν να αναπαραχθούν για χρήση σε διάφορα πεδία εφαρμογών προσφέροντας έτσι ένα «κουτί» που ασχολείται με τη διαχείριση των δεδομένων παρέχοντας κάποιες βασικές λειτουργίες.

Μέσω αυτής της διπλωματικής εργασίας παρέχω τη δυνατότητα διαχείρισης των δεδομένων αυτών που συλλέγονται μέσω πλατφόρμων μικροελεγκτών. Αυτό περιλαμβάνει την προβολή των μετρήσεων από διάφορους αισθητήρες και την αλλαγή κάποιων ρυθμίσεων σχετικών με τη συλλογή δεδομένων από αυτούς τους αισθητήρες. Οι ρυθμίσεις αυτές μπορεί να είναι η συχνότητα συλλογής μετρήσεων από κάποιο αισθητήρα ή μικροελεγκτή και η απενεργοποίησή τους.

Μερικές βασικές έννοιες των οποίων η γνώση απαιτείται για τη συγκεκριμένη διπλωματική εργασία περιγράφονται παρακάτω:

Εφαρμογές επίγνωσης συγκειμένου:

Η ιδέα του συγκειμένου (context) εισάχθηκε γύρω στην έναρξη της χιλιετίας και συγκεκριμένα το 1999 περιγράφοντας «*οποιαδήποτε πληροφορία που μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα είναι ένα άτομο, τοποθεσία ή αντικείμενο που θεωρείται σχετικό στην αλληλεπίδραση μεταξύ ενός χρήστη και μιας εφαρμογής, συμπεριλαμβανομένων του χρήστη και της εφαρμογής*»[9].

Οι εφαρμογές επίγνωσης συγκειμένου λαμβάνουν υπόψη αυτές τις πληροφορίες έτσι ώστε να προσαρμόσουν την λειτουργικότητά τους στο συγκείμενο της χρήσης τους.

Πλατφόρμες μικροελεγκτών (Single-board microcontrollers):

Με τον όρο πλατφόρμα μικροελεγκτή εννοούμε ένα μικροελεγκτή ο οποίος είναι χτισμένος πάνω σε μια πλακέτα κυκλώματος. Αυτή η πλακέτα παρέχει όλες τις απαραίτητες δυνατότητες ελέγχου όπως εισόδους/εξόδους, μνήμη τυχαίας προσπέλασης (RAM) , μικροεπεξεργαστή και μνήμη αποθήκευσης προγραμμάτων.

Οι επικρατέστερες πλατφόρμες μικροελεγκτών στις μέρες μας είναι το Arduino και το Raspberry Pi που χρησιμοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας.

Raspberry Pi:

Ο όρος Raspberry Pi αναφέρεται σε μια σειρά πλατφόρμων μικροελεγκτών που αναπτύχθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation με πρωταρχικό στόχο τη χρήση τους στην προώθηση της διδασκαλίας βασικών εννοιών της Πληροφορικής στα σχολεία. Το Raspberry Pi είναι σχεδιασμένο για να λειτουργεί ως μικρής ισχύος ηλεκτρονικός υπολογιστής παρέχοντας 512 mb RAM και θύρες USB, HDMI, και Ethernet δίνοντας έτσι τη δυνατότητα διασύνδεσής του με το διαδίκτυο και άλλες εξωτερικές συσκευές.

Επίσης, το Raspberry Pi παρέχει τη δυνατότητα διασύνδεσης με το περιβάλλον αφού πάνω σε αυτό μπορεί να ενσωματωθεί μια πληθώρα αισθητήρων και ενεργοποιητών που μπορούν να προγραμματιστούν μέσω Python.

Android Software Development:

Η ανάπτυξη εφαρμογών Android αναφέρεται στη διαδικασία στην οποία γίνεται η δημιουργία εφαρμογών για κινητές συσκευές που προορίζονται για το λειτουργικό σύστημα Android. Κύρια γλώσσα ανάπτυξης τέτοιου είδους εφαρμογών είναι η Java και οι εφαρμογές αυτές χρησιμοποιούν το Android SDK (Software development kit).

Android SDK:

Με τον όρο Android Software Development Tools εννοούμε μια σειρά από εργαλεία που βοηθούν στην ανάπτυξη εφαρμογών Android. Πρωτοεμφανίστηκε τον Οκτώβρη του 2009 από την Google και είναι γραμμένο στη γλώσσα προγραμματισμού Java. Η συλλογή από εργαλεία περιλαμβάνει debugger, βιβλιοθήκες που υποβοηθούν στην ανάπτυξη εφαρμογών, πηγαίο κώδικα, τεκμηρίωση, εγχειρίδια χρήσης και προσομοιωτή. Μέχρι το τέλος του 2014 το επίσημο περιβάλλον ανάπτυξης ήταν το eClipse χρησιμοποιώντας το plug-in Android Development Tools (ADT) . Από την αρχή του 2015, το επίσημο εργαλείο ανάπτυξης Android εφαρμογών είναι το Android – studio που υλοποιήθηκε από την Google και την IntelliJ. Οι Android εφαρμογές συσκευάζονται σε αρχεία που έχουν την κατάληξη .apk και ο εκτελέσιμος κώδικας ονομάζεται Dalvik executable αφού τρέχει σε Dalvik virtual machine.

Υπηρεσίες Διαδικτύου / RESTful υπηρεσίες διαδικτύου:

«Είναι αυτό-περιγραφικές, ανεξάρτητες, αρθρωτές εφαρμογές που μπορούν να δημοσιευτούν, να εντοπιστούν και να κληθούν από το διαδίκτυο. Οι υπηρεσίες διαδικτύου εκτελούν συναρτήσεις, που μπορεί να είναι οτιδήποτε από ένα απλό αίτημα έως μια περίπλοκη επιχειρησιακή διαδικασία. Μόλις μια υπηρεσία διαδικτύου αναπτυχθεί, άλλες εφαρμογές (και άλλες υπηρεσίες διαδικτύου) μπορούν να την αναζητήσουν και να την καλέσουν.» - IBM. Με τον όρο REST (Representational State Transfer) αναφερόμαστε σε ένα αρχιτεκτονικό πρότυπο που καθοδηγεί την ανάπτυξη υπηρεσιών διαδικτύου καθορίζοντας κανόνες και οδηγίες στο σχεδιασμό επεκτάσιμων υπηρεσιών διαδικτύου.

Αισθητήρες:

Οι αισθητήρες είναι ξεχωριστές συσκευές ή περίπλοκες κατασκευές των οποίων η βασική λειτουργία είναι η ανίχνευση ενός σήματος ή μιας διέγερσης και η παραγωγή μιας μετρήσιμης εξόδου. Συνήθως, η έξοδος ενός αισθητήρα είναι ένα αναλογικό ή ψηφιακό σήμα. Υπάρχουν πολλά είδη αισθητήρων που ανιχνεύουν ένα ευρύ φάσμα τιμών με διαφορετική ακρίβεια. Μερικά παραδείγματα αισθητήρων είναι θερμότητας, φωτός, υγρασίας, κίνησης, ήχου κλπ.

Internet of Things (IoT):

Το Internet of Things (IoT) είναι ένα σενάριο στο οποίο όλα τα αντικείμενα, ζώα ή άνθρωποι μπορούν να μεταφέρουν δεδομένα σε ένα δίκτυο έχοντας μοναδικό τρόπο ταυτοποίησης, επικοινωνώντας έτσι με το μεταξύ τους και με το δίκτυο χωρίς την άμεση επέμβαση των ανθρώπων[22].

1.2 Στόχος Διπλωματικής Εργασίας

Ένας από τους κύριους στόχους της ατομικής διπλωματικής εργασίας ήταν ο συνδυασμός πλατφόρμων μικροελεγκτών όπως είναι το Arduino και το Raspberry Pi με μια εφαρμογή για κινητές συσκευές Android. Αυτό οδήγησε στην υλοποίηση της εφαρμογής SensoMan.

Το κύριο πρόβλημα που προσπαθεί να λύσει η εφαρμογή SensoMan είναι η διαχείριση αισθητήρων από απομακρυσμένη τοποθεσία. Οι αισθητήρες που χρησιμοποιούνται μπορεί να βρίσκονται κατανεμημένοι σε διαφορετικές πλατφόρμες μικροελεγκτών. Συνεπώς δεν υπάρχει ένας ενιαίος τρόπος πρόσβασης σε αυτούς τους αισθητήρες. Αυτό που προσπαθεί να επιτύχει η εφαρμογή είναι η παροχή δυνατότητας στο χρήστη να έχει υπό την επίβλεψη του μια πληθώρα αισθητήρων που θα μπορεί να αλλάζει τις διάφορες παραμέτρους συλλογής των δεδομένων του.

Οι δυνατότητες που πρέπει να δίνονται στο χρήστη περιλαμβάνουν την ανά πάσα στιγμή λήψη των τελευταίων ή ιστορικών μετρήσεων των αισθητήρων τους οποίους διαχειρίζεται και τη δυνατότητα αλλαγής ρυθμίσεων σχετικά με τους αισθητήρες όπως την απενεργοποίηση ενός αισθητήρα και την αλλαγή της συχνότητας που μαζεύει δεδομένα ένας αισθητήρας.

1.2.1 Προηγούμενες εργασίες

Σχετικά με τη διαχείριση των πλατφόρμων μικροελεγκτών υπάρχουν κάποιες εφαρμογές που είναι είτε σε πρώιμα στάδια ή δεν προσφέρουν στους τελικούς χρήστες τους αυτά που υπόσχονται.
Παραδείγματα τέτοιων εφαρμογών είναι το Arduino Manager [10]το οποίο έχει να κάνει με τη διαχείριση της πλατφόρμας μικροελεγκτή Arduino το οποίο εκτός από το ότι αφορά αποκλειστικά το Arduino, σύμφωνα με τις κριτικές των χρηστών της εφαρμογής δεν εκπληρώνει τις υποσχέσεις του στους τελικούς χρήστες.
Ακόμα ένα παράδειγμα είναι αυτό εφαρμογών όπως το ConnectBot [11] ή το VNC Viewer/ Server [12] που προσφέρουν σύνδεση με το Raspberry Pi μέσω SSH. Οι προαναφερθείσες εφαρμογές παρόλο που έμμεσα δίνουν τη δυνατότητα διαχείρισης του μικροελεγκτή με τη χρήση εντολών στο κέλυφος του, δεν μπορούν άμεσα να διαχειριστούν τους αισθητήρες που είναι συνδεδεμένοι πάνω στο μικροελεγκτή. Επιπλέον, οι πιο πάνω εφαρμογές δεν προσφέρουν τη δυνατότητα σύνδεσης με μικροελεγκτές Arduino λόγω του τρόπου λειτουργίας του μικροελεγκτή αυτού.

1.2.2 Προσέγγιση

Η προσέγγιση που προσφέρει αυτή η ατομική διπλωματική εργασία εστιάζεται στη λήψη πληροφοριών των αισθητήρων και διαχείριση των μικροελεγκτών μέσω REST services από την εφαρμογή για κινητές συσκευές.

Λόγω του ότι και οι δύο πλατφόρμες μικροελεγκτών έχουν πρόσβαση στο διαδίκτυο, υπάρχει η δυνατότητα υλοποίησης μιας υπηρεσίας διαδικτύου που δρα ως ενδιάμεσο επίπεδο για την επικοινωνία πλατφόρμας μικροελεγκτή και εφαρμογής.

Συγκεκριμένα, οι πλατφόρμες μικροελεγκτών μετά από τη λήψη μετρήσεων από τους αισθητήρες που βρίσκονται συνδεδεμένοι σε αυτές, στέλνουν αυτά τα δεδομένα σε μια βάση δεδομένων που βρίσκεται στο διαδίκτυο. Αυτό γίνεται με τη βοήθεια HTTP requests στο REST API που προσφέρει η υπηρεσία διαδικτύου. Στη συνέχεια, η κινητή εφαρμογή Android λαμβάνει τα δεδομένα μέσω HTTP requests από το REST API.

Το ίδιο συμβαίνει και στην περίπτωση αλλαγής ρυθμίσεων. Η εφαρμογή στέλνει HTTP request στο REST API και η υπηρεσία διαδικτύου είναι υπεύθυνη για τη διεκπεραίωση των εντολών.

Με αυτό τον τρόπο επιτυγχάνεται επίσης μια αφαιρετικότητα και ευελιξία επιτρέποντας στο χρήστη να προσθέσει ανά πάσα στιγμή νέους αισθητήρες και μικροελεγκτές στο σύστημα και να γίνεται αυτόματα η ενσωμάτωσή τους και η προβολή των μετρήσεων τους από την εφαρμογή.

1.3 Δομή διπλωματικής εργασίας

Η συγκεκριμένη ατομική διπλωματική εργασία αποτελείται από επτά κεφάλαια.

Στο πρώτο κεφάλαιο παρουσιάζεται μια εισαγωγή στο θέμα της ατομικής διπλωματικής εργασίας και επεξήγηση βασικών όρων που χρησιμοποιούνται σε όλη την έκταση του κειμένου. Ακόμα, γίνεται αναφορά στο στόχο της ατομικής διπλωματικής εργασίας, σε παρόμοιες προσεγγίσεις του συγκεκριμένου στόχου και η προσέγγιση που αποφασίστηκε να ακολουθηθεί.

Στο δεύτερο κεφάλαιο της ατομικής διπλωματικής εργασίας γίνεται αναλυτική αναφορά στην πλατφόρμα μικροελεγκτή που χρησιμοποιήθηκε για τη συγκεκριμένη Α.Δ.Ε, Raspberry Pi και στους αισθητήρες που βρίσκονται ενωμένοι στο συγκεκριμένο μικροελεγκτή. Επιπλέον, γίνεται αναφορά στις γλώσσες προγραμματισμού και άλλες τεχνολογίες που χρησιμοποιήθηκαν για τη συλλογή δεδομένων από τους συγκεκριμένους αισθητήρες.

Στο τρίτο κεφάλαιο γίνεται αναφορά στις απαιτήσεις της εφαρμογής που αναπτύχθηκε και στο σχεδιασμό της υλοποίησης της εφαρμογής. Παρατίθεται αρχικά μια ανάλυση των απαιτήσεων της συγκεκριμένης εφαρμογής με τη βοήθεια σεναρίων και διαγραμμάτων χρήσης. Επιπλέον, εξηγείται ο σχεδιασμός της εφαρμογής με τη χρήση διαγραμμάτων UML.

Στο τέταρτο κεφάλαιο γίνεται αναλυτική αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής για κινητές συσκευές Android. Συγκεκριμένα, γίνεται αναφορά στο Android SDK, στις βιβλιοθήκες που χρησιμοποιήθηκαν και στις υπηρεσίες διαδικτύου που χρησιμοποιεί η εφαρμογή. Τέλος, γίνεται αναφορά στην υπηρεσία Anyplace την οποία χρησιμοποιεί η συγκεκριμένη εφαρμογή.

Στο πέμπτο κεφάλαιο περιγράφεται αναλυτικά η κάθε οθόνη της εφαρμογής για κινητές συσκευές και γίνεται μια επεξήγηση των διαθέσιμων ρυθμίσεων που προσφέρει η εφαρμογή.

Στο έκτο κεφάλαιο γίνεται ανάλυση της μεθόδου που χρησιμοποιήθηκε για την αξιολόγηση της πιο πάνω εφαρμογής για κινητές συσκευές και εξαγωγή σχετικών αποτελεσμάτων.

Στο έβδομο κεφάλαιο γίνεται αναφορά στα συμπεράσματα που εξήχθηκαν μετά την διεκπεραίωση της αξιολόγησης της κινητής εφαρμογής καθώς και στις δυνατότητες επέκτασης της κινητής εφαρμογής και γενικά όλου του συστήματος SensoMan.

2 Πλατφόρμα Μικροελεγκτή Raspberry Pi

2.1 Περιγραφή

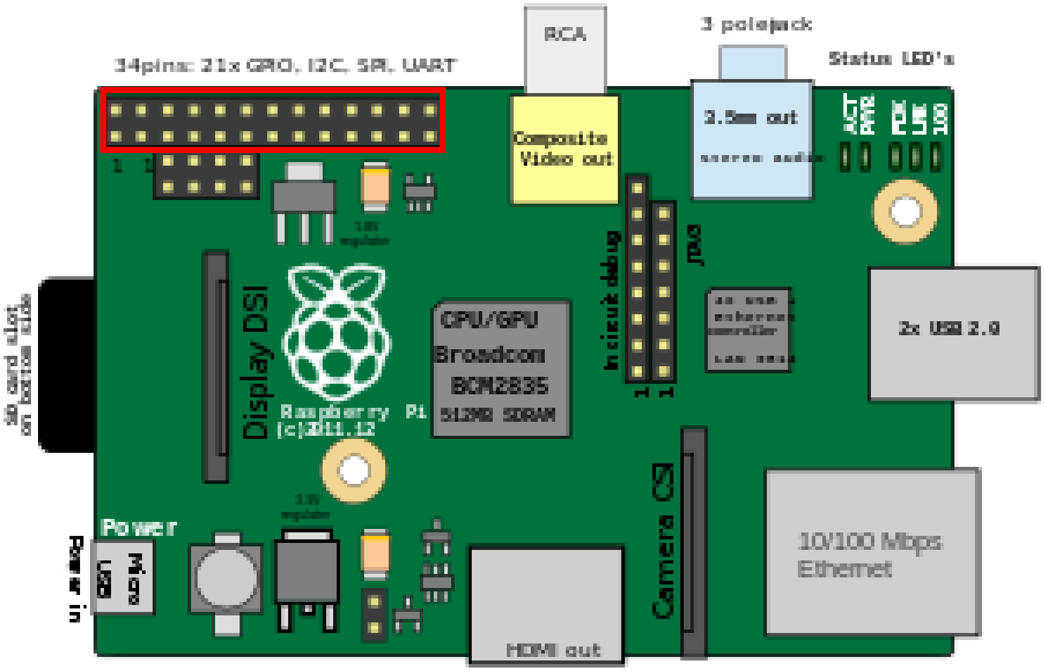
Το Raspberry Pi, είναι ένας χαμηλού κόστους ηλεκτρονικός υπολογιστής σε μέγεθος πιστωτικής κάρτας, που δημιουργήθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi foundation [21]. Ο κύριος σκοπός δημιουργίας του Raspberry Pi ήταν η διδασκαλία της επιστήμης της Πληροφορικής στα σχολεία.

Μέχρι τώρα έχουν υλοποιηθεί τέσσερα μοντέλα Raspberry Pi με πρώτο το Raspberry Pi 1 model A που ακολουθείται από το Raspberry Pi 1 model B, Raspberry Pi 1 model B+ και Raspberry Pi 2 Model B.

Για τους σκοπούς της ατομικής διπλωματικής εργασίας χρησιμοποιήθηκε το Raspberry Pi 1 model B.

2.1.1 Raspberry Pi 1 model B

Το Raspberry Pi model B είναι μια ψηλότερων προδιαγραφών παραλλαγή του αρχικού Raspberry Pi. Περιέχει 512MB RAM, 2 USB ports και θύρα Ethernet .



Εικόνα 1: Raspberry Pi Model B

Πέραν αυτού, το Raspberry Pi περιέχει 21 GPIO pins τα οποία χρησιμοποιούνται για τη σύνδεση του με εξωτερικά κυκλώματα. Επιπρόσθετα παρέχονται πολλές δυνατότητες διασύνδεσης με περιφερειακές συσκευές εισόδου/εξόδου όπως ποντίκι, πληκτρολόγιο, οθόνη (μέσω HDMI cable) , camera κλπ. Έχοντας το δικό του επεξεργαστή λειτουργεί αυτόνομα ως ηλεκτρονικός υπολογιστής.

Αξίζει να σημειωθεί ότι τα pins που βρίσκονται στο GPIO module διαβάζουν μόνο ψηφιακό σήμα.

Ακόμα μία αξιοσημείωτη δυνατότητα που παρέχει το Raspberry Pi είναι το ότι επιτρέπει τη σύνδεση με Arduino μέσω καλωδίου USB και η επικοινωνία και έλεγχος του μέσω μιας γλώσσας προγραμματισμού π.χ. Python. Αυτό περιγράφει μια σημαντική ιδιότητα του Raspberry Pi να λειτουργεί με το Arduino ως master-slave δίνοντας εντολές στο Arduino και λαμβάνοντας μετρήσεις και αποτελέσματα. Επιπλέον, στο

Raspberry Pi θα μπορούσε να εγκατασταθεί το πρόγραμμα που συνδέει το Arduino με ηλεκτρονικό υπολογιστή και να λειτουργεί ως μέθοδος φόρτωσης προγραμμάτων πάνω στο Raspberry Pi.

2.1.2 Λειτουργικό Σύστημα

Υπάρχουν πολυάριθμα λειτουργικά συστήματα που μπορούν να εγκατασταθούν στην πλατφόρμα μικροελεγκτή Raspberry Pi με τα πιο δημοφιλή από αυτά να είναι Linux διανομές.

Η υπερισχύουσα επιλογή είναι το Raspbian το οποίο βασίζεται στη Linux διανομή Debian. Το Raspbian έρχεται με χρήσιμα προγράμματα όπως το IDLE, ένα επεξεργαστή κειμένου για Python και το Scratch ένα σύστημα για δημιουργία παιχνιδιών με σκοπό την εκμάθηση προγραμματισμού.

Ακόμα μια επιλογή που προορίζεται για πιο εξειδικευμένες μετρήσεις από αισθητήρες και είναι περισσότερο προσανατολισμένη στο υλικό του Raspberry Pi είναι το Occidentalis που

2.2 Γλώσσες προγραμματισμού

Η ιδιότητα του Raspberry Pi να λειτουργεί ως αυτόνομος υπολογιστής παρέχει τη δυνατότητα προγραμματισμού σε πολυάριθμες γλώσσες προγραμματισμού.

Αρχικά, λόγω του ότι το λειτουργικό σύστημα που εγκαταστάθηκε στο Raspberry Pi είναι μια διανομή του Linux παρέχεται η δυνατότητα μέσω του λειτουργικού συστήματος ο προγραμματισμός σε bash.

Επιπλέον, η κύρια γλώσσα που χρησιμοποιείται στο Raspberry Pi είναι η Python αφού παρέχει δυνατότητες πρόσβασης στα GPIO pins που βρίσκονται στο Raspberry Pi. Η Python μπορεί να χρησιμοποιηθεί και για πρόσβαση στα USB και Ethernet ports.

Όπως προαναφέρθηκε, λόγω των δυνατοτήτων του, το Raspberry Pi μπορεί να προγραμματιστεί με μεγάλο αριθμό γλωσσών προγραμματισμού όπως για παράδειγμα Java, C, C++ κλπ.

2.3 Αισθητήρες

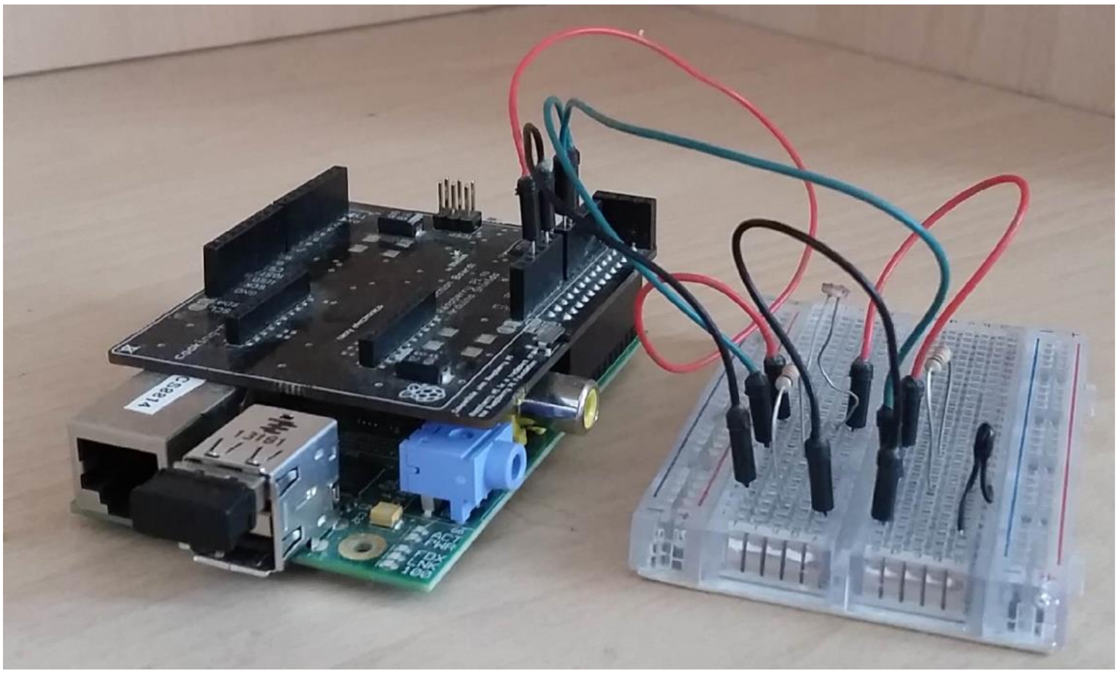
Για τους σκοπούς της ατομικής διπλωματικής εργασίας χρησιμοποιήθηκαν δύο αισθητήρες για συλλογή δεδομένων. Οι αισθητήρες αυτοί είναι ένας φωτοαντιστάτης και ένας θερμοαντιστάτης (thermistor) γενικού τύπου. Σημειώνεται ότι οι δύο πιο πάνω αισθητήρες δεν προσφέρουν τη μέγιστη δυνατή ακρίβεια αλλά είναι ένα καλό σημείο εκκίνησης για εξοικείωση με τον τρόπο συλλογής δεδομένων από το Raspberry Pi.
Παρόλα αυτά, στο Raspberry Pi μπορεί να συνδεθεί μία πληθώρα από αισθητήρες με διαφορετικά χαρακτηριστικά ο καθένας και διαφορετικά σήματα εξόδου.

2.3.1 Φωτοκύτταρο – LDR αισθητήρας:

Ο αισθητήρας που χρησιμοποιήθηκε για μέτρηση του φωτός είναι ο PDV-P9203. Το συγκεκριμένο είδος αισθητήρα είναι σχεδιασμένο για να αισθάνεται το φώς από 400 σε 700 nm. Αυτοί οι φωτοαντιστάτες είναι διαθέσιμοι σε ευρεία κλίμακα αντιστάσεων. Ο συγκεκριμένος αισθητήρας έχει αναλογική έξοδο. Περισσότερες πληροφορίες για το συγκεκριμένο φωτοαντιστάτη καθώς και το datasheet για το συγκεκριμένο αισθητήρα βρίσκονται στη βιβλιογραφία [1].

2.3.2 Θερμοαντιστάτης – NTC αισθητήρας:

Ο αισθητήρας που χρησιμοποιήθηκε για μέτρηση της θερμοκρασίας είναι ο NTC . Το συγκεκριμένο είδος αισθητήρα δρα ως θερμοαντιστάτης με μηδενική αντίσταση στους 25 ºC. Ο συγκεκριμένος αισθητήρας έχει αναλογική έξοδο. Περισσότερες πληροφορίες για το συγκεκριμένο θερμοαντιστάτη καθώς και το datasheet για το συγκεκριμένο αισθητήρα βρίσκονται στη βιβλιογραφία [2].



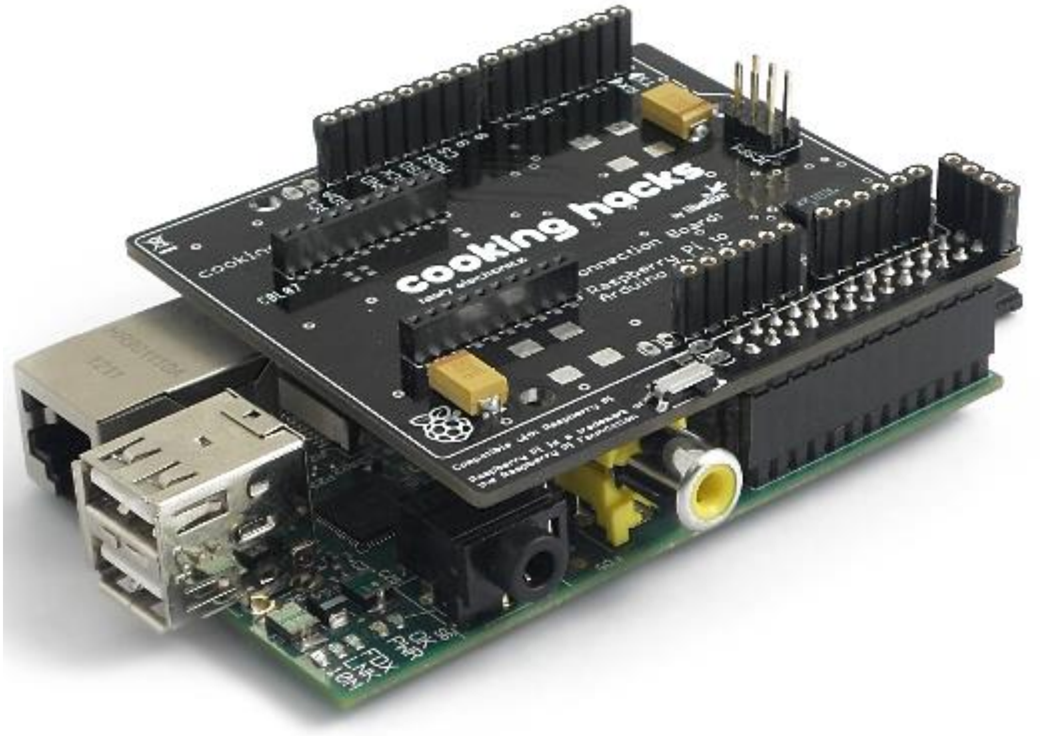
Εικόνα 2: Οι αισθητήρες συνδεδεμένοι στο Raspberry Pi χρησιμοποιώντας το Raspberry Pi to Arduino shield

Στην πιο πάνω εικόνα (Εικόνα 2) φαίνονται οι δύο αισθητήρες που βρίσκονται συνδεδεμένοι στο Raspberry Pi. Όπως φαίνεται από την εικόνα, χρησιμοποιείται το Raspberry Pi to Arduino module και η έξοδος των δύο αισθητήρων είναι συνδεδεμένη στα αναλογικά pins που προσφέρει το shield. Αυτό διευκολύνει τη λήψη μετρήσεων αφού μετατρέπει με τη χρήση μιας βιβλιοθήκης το αναλογικό σήμα των αισθητήρων σε μορφή που μπορεί να αναγνωριστεί από το Raspberry Pi. Η σύνδεση των πιο πάνω αισθητήρων χωρίς το συγκεκριμένο shield θα απαιτούσε επιπλέον εξαρτήματα όπως πυκνωτές για μετατροπή του αναλογικού τους σήματος σε ψηφιακό.

Αξίζει επίσης να σημειωθεί ότι το κύκλωμα που χρησιμοποιήθηκε για την ένωση των δύο αισθητήρων στο Raspberry Pi δεν διαφέρει από το κύκλωμα που χρησιμοποιήθηκε για την ένωση των ίδιων αισθητήρων στο Arduino.

2.4 Raspberry Pi to Arduino shield

Η ιδέα πίσω από τα Raspberry Pi to Arduino shields είναι να το να επιτρέψει οποιαδήποτε από τα shields, boards και connection modules που χρησιμοποιούνται στο Arduino να χρησιμοποιηθούν στο Raspberry Pi. Συγκεκριμένα, μετατρέπει τα pins εισόδου και εξόδου του Raspberry Pi σε αυτά του Arduino διατηρώντας ανέπαφες όλες τις υπόλοιπες λειτουργίες του Raspberry Pi.



Εικόνα 3: Raspberry Pi to Arduino Shield

Επιπλέον, για ολοκλήρωση της συμβατότητας, η εταιρία που δημιούργησε το συγκεκριμένο module υλοποίησε επίσης τη βιβλιοθήκη arduPi η οποία επιτρέπει τη χρήση του κώδικα που χρησιμοποιείται στο Arduino να χρησιμοποιηθεί και στο Raspberry Pi. Αυτή η βιβλιοθήκη, που είναι γραμμένη σε C++ υλοποιεί συναρτήσεις που επιτρέπουν τον έλεγχο της εισόδου/εξόδου του Raspberry Pi σαν Arduino είσοδο/έξοδο συμπεριλαμβανομένων των αναλογικών εισόδων/εξόδων που έχει το Arduino. Το πιο πάνω επιτυγχάνεται με υλοποιημένες συναρτήσεις της βιβλιοθήκης που αναλαμβάνουν τη μετατροπή και τον έλεγχο των σημάτων.

Οι δυνατότητες που παρέχουν το συγκεκριμένο module και βιβλιοθήκη στάθηκαν πολύ βοηθητικές στη διαδικασία λήψης μετρήσεων από τους αισθητήρες που ήταν συνδεδεμένοι στο Raspberry Pi αφού ανέλαβε τη μετατροπή του αναλογικού σήματος από τους αισθητήρες σε ψηφιακό σήμα που αναγνωρίζουν τα GPIO pins του Raspberry Pi. Χρησιμοποιώντας την πιο πάνω μέθοδο, εξαλείφτηκε η ανάγκη χρήσης επιπλέων κυκλωμάτων και modules και διευκόλυνε τη συγγραφή του κώδικα αφού πλέον ο κώδικας που δημιουργήθηκε είναι γραμμένος σε C++.

Συμπερασματικά, για τη συλλογή μετρήσεων από τους αισθητήρες χρησιμοποιήθηκε η συγκεκριμένη μέθοδος και ουσιαστικά για το κομμάτι συλλογής δεδομένων από τους αισθητήρες το Raspberry Pi χρησιμοποιήθηκε ως Arduino.

Περισσότερες πληροφορίες για το συγκεκριμένο module και τη βιβλιοθήκη arduPi βρίσκονται στο σχετικό documentation μαζί με παραδείγματα κώδικα που ενσωματώνει τη συγκεκριμένη βιβλιοθήκη για δημιουργία προγραμμάτων στο Raspberry Pi [3]. Ένα παράδειγμα κώδικα που χρησιμοποιείται για τη λήψη μετρήσεων από τους αισθητήρες φαίνεται πιο κάτω (Σχήμα 1).

```
int photocellPin = 0; // the cell and 10K pulldown are connected
to a0

    int photocellReading; // the analog reading from the sensor
divider

    int LEDpin = 11; // connect Red LED to pin 11 (PWM pin)
(this is not needed to use the sensor but if you have one laying
around it could add a cool affect! This can be bought at your
local Radio Shack.)

    int LEDbrightness; //

    void setup(void) {

    Serial.begin(9600);    // We'll send debugging information
via the Serial monitor so if you can read it   without using a
led



    }

    void loop(void) {

    photocellReading = analogRead(photocellPin);

    Serial.print("Analog reading = ");

    Serial.println(photocellReading); // the raw analog reading

    // LED gets brighter the darker it is at the sensor

    // that means we have to -invert- the reading from 0-1023
back to 1023-0

    photocellReading = 1023 - photocellReading;    //now we have
to map 0-1023 to 0-255 since thats the          range
analogWrite uses

    LEDbrightness = map(photocellReading, 0, 1023, 0, 255);

    analogWrite(LEDpin, LEDbrightness);

    delay(100);

    }
```

Σχήμα 1 : Κώδικας για ανάγνωση σημάτων από αισθητήρες στο Arduino

3 Κινητή εφαρμογή SensoMan

Σε αυτό το κεφάλαιο γίνεται λεπτομερής αναφορά στην ανάλυση των απαιτήσεων της εφαρμογής SensoMan καθώς και στο σχεδιασμό και τις λειτουργίες της εφαρμογής. Η ανάλυση των απαιτήσεων και ο σχεδιασμός της εφαρμογής γίνεται με τη βοήθεια σχεδιαγραμμάτων UML. Στη συνέχεια, παρουσιάζονται όλες οι οθόνες τις εφαρμογής και επεξηγείται αναλυτικά η λειτουργία της κάθε οθόνης στην εφαρμογή.

3.1 Ανάλυση απαιτήσεων

Όπως προαναφέρθηκε, οι κύριες απαιτήσεις της εφαρμογής είναι:

1) η διαχείριση των αισθητήρων από απομακρυσμένη τοποθεσία και,
2) η προβολή των μετρήσεων από τους αισθητήρες από απομακρυσμένη τοποθεσία.

Για καλύτερη ανάλυση των απαιτήσεων της εφαρμογής, χρησιμοποιήθηκαν δύο μέθοδοι εξαγωγής των απαιτήσεων. Οι μέθοδοι αυτοί περιλαμβάνουν τα διαγράμματα περιπτώσεων χρήσης της UML (Use Case diagrams) και σενάρια χρήσης της εφαρμογής.

Σε αυτή την ενότητα εξηγούνται αναλυτικά οι απαιτήσεις του συστήματος με τη βοήθεια διαγραμμάτων UML και σεναρίων χρήσης.

3.1.1 Περιπτώσεις χρήσης

Τα διαγράμματα χρήσης περιγράφουν περιληπτικά όλες τις λειτουργίες του συστήματος και από ποιες ομάδες χρηστών εκτελούνται οι συγκεκριμένες λειτουργίες. Το διάγραμμα χρήσης της εφαρμογής παρουσιάζεται και επεξηγείται πιο κάτω.

Εικόνα 4: Διάγραμμα χρήσης SensoMan

Όπως φαίνεται και από το πιο πάνω διάγραμμα, στην εφαρμογή υπάρχουν τρείς μεγάλες κατηγορίες χρηστών.

- **Απλός χρήστης (Simple user):** Οι χρήστες αυτοί χρησιμοποιούν τη συγκεκριμένη εφαρμογή μόνο για προβολή των μετρήσεων από τους αισθητήρες.
- **Ιδιοκτήτης αισθητήρων (Sensor owner):** Οι χρήστες αυτοί μπορούν να χρησιμοποιήσουν την εφαρμογή για προβολή των μετρήσεων των αισθητήρων τους οποίους κατέχουν και για αλλαγή των ρυθμίσεων των συγκεκριμένων αισθητήρων/ πλατφόρμων μικροελεγκτών που έχουν υπό την κατοχή τους.
- **Διαχειριστής (Root User/Admin):** Οι χρήστες αυτοί μπορούν να χρησιμοποιήσουν την εφαρμογή για προβολή των στοιχείων των αισθητήρων και την αλλαγή των ρυθμίσεων όλων των αισθητήρων στο σύστημα.

3.1.2 Σενάριο χρήσης του συστήματος

Για εύκολη εξαγωγή των απαιτήσεων της κάθε ομάδας χρηστών από το σύστημα δημιουργήθηκαν πιθανά σενάρια χρήσης του συστήματος για τις τρείς κατηγορίες χρηστών. Τίθεται το παράδειγμα της χρήσης του συστήματος για παρακολούθηση και διαχείριση των αισθητήρων που αποτελούν ένα σύστημα ασφαλείας σε ένα μουσείο. Λόγω του μεγάλου αριθμού αισθητήρων που υπάρχουν στο μουσείο, ο διαχειριστής έχει διαχωρίσει σε δύο άτομα τη διαχείριση των αισθητήρων. Επίσης, υπάρχει ένα άτομο το οποίο είναι υπεύθυνο για τη στενή παρακολούθηση όλων των αισθητήρων και ενημέρωση του διαχειριστή ή των ιδιοκτητών των αισθητήρων για τυχόν σφάλματα ή μετρήσεις που υποδεικνύουν διάρρηξη στο μουσείο ή κάποια καταστροφή.

Επιστάτης μουσείου - Απλός χρήστης: Το άτομο αυτό δεν είναι υπεύθυνο για τη διαχείριση των αισθητήρων αλλά για την παρακολούθηση των μετρήσεων του κάθε αισθητήρα για ανίχνευση τυχών δυσλειτουργίας στους αισθητήρες. Επίσης, το άτομο αυτό είναι υπεύθυνο για την παρακολούθηση των τελευταίων μετρήσεων από τους αισθητήρες ώστε να αντιληφθεί αν υπάρχει περίπτωση κακόβουλης ενέργειας σε κάποιο από τα εκθέματα του μουσείο. Ο επιστάτης, λαμβάνει νέα αποτελέσματα από την εφαρμογή κάθε δύο λεπτά και στη συνέχεια συγκρίνει το νέο αποτέλεσμα με τα προηγούμενα.

Υπεύθυνοι συντήρησης συστήματος – Ιδιοκτήτες αισθητήρων: Τα άτομα αυτά είναι υπεύθυνα για τη σωστή λειτουργία των αισθητήρων. Όταν ενημερώνονται από τον επιστάτη για περίεργες μετρήσεις από το σύστημα συγκρίνουν τις μετρήσεις των αισθητήρων κατά τη διάρκεια ενός χρονικού διαστήματος. Αν αποδειχθεί ότι ένας αισθητήρας δυσλειτουργεί, για παράδειγμα ο αισθητήρας θερμότητας δίνει υπερβολικά ψηλές θερμοκρασίες για ένα μεγάλο χρονικό διάστημα, τότε ο ιδιοκτήτης του συγκεκριμένου αισθητήρα μπορεί να τον απενεργοποιήσει έτσι ώστε να μην φαίνονται οι λανθασμένες τιμές στο σύστημα. Στη συνέχεια, ο ιδιοκτήτης των αισθητήρων μπορεί να αλλάξει τον ελαττωματικό αισθητήρα και να προσθέσει ένα νέο αισθητήρα στο σύστημα για λήψη των νέων τιμών.

Ακόμα, ο ιδιοκτήτης των αισθητήρων μπορεί να αλλάζει τη συχνότητα της ανάκτησης δεδομένων από τους αισθητήρες. Για παράδειγμα, σε ώρες που ο κίνδυνος διάρρηξης είναι μεγάλος, θα πρέπει να μπορεί να μεγαλώνει τη συχνότητα ανάκτησης δεδομένων για γρηγορότερο εντοπισμό παράβασης.

Διαχειριστής συστήματος : Το άτομο αυτό θα πρέπει να είναι σε θέση να εκτελεί όλες τις λειτουργίες του συστήματος. Θα τίθεται ως ιδιοκτήτης όλων των αισθητήρων και θα μπορεί ανά πάσα στιγμή να δει τις τελευταίες αλλά και παλαιότερες μετρήσεις από όλους τους αισθητήρες. Επιπλέον, είναι σε θέση να αλλάζει τη συχνότητα όλων των αισθητήρων αλλά και να απενεργοποιεί αισθητήρες. Εκτός αυτού, ο διαχειριστής του συστήματος θα πρέπει να είναι σε θέση να προσθέτει νέους αισθητήρες αλλά και πλατφόρμες μικροελεγκτών στο σύστημα.

3.1.3 Λειτουργικές απαιτήσεις συστήματος

Από το πιο πάνω σενάριο και το διάγραμμα χρήσης εξήχθηκαν οι πιο κάτω λειτουργίες που θα πρέπει να υλοποιεί το σύστημα διαχείρισης των αισθητήρων:

- Είσοδος στο σύστημα για διαφορετικές ομάδες χρηστών

Είναι απαραίτητος ο καθορισμός μιας μεθόδου αναγνώρισης του τύπου χρήστη που ζητά πρόσβαση στο σύστημα καθώς διαφορετικοί χρήστες θα έχουν διαφορετικές δυνατότητες χρήσης της εφαρμογής. Το σύστημα θα πρέπει να δέχεται το συνθηματικό

του χρήστη και να αναγνωρίζει αν πρόκειται για απλό χρήστη, ιδιοκτήτη αισθητήρων ή διαχειριστή και να ενεργοποιεί / απενεργοποιεί κάποιες λειτουργίες ανάλογα. Επίσης , ανάλογα με τον τύπο του χρήστη αλλά και τα στοιχεία εισαγωγής του στο σύστημα, επιστρέφονται μόνο τα δεδομένα που τον αφορούν (προβολή μόνο των δεδομένων των αισθητήρων τους οποίους διαχειρίζεται ο συγκεκριμένος χρήστης).
Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Απλός χρήστης, Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Προβολή τελευταίων μετρήσεων από αισθητήρες

Άλλη μια βασική απαίτηση που θα πρέπει να υλοποιεί το σύστημα είναι να δίνει τη δυνατότητα σε όλους τους χρήστες να βλέπουν ανά πάσα στιγμή τις τελευταίες μετρήσεις από όλους τους αισθητήρες για τους οποίους έχουν δικαίωμα να βλέπουν τις τιμές. Αυτό περιλαμβάνει το χρόνο τον οποίο λήφθηκε η συγκεκριμένη μέτρηση, η τιμή του αισθητήρα τη συγκεκριμένη χρονική στιγμή για πιο αισθητήρα πρόκειται αλλά και πληροφορίες σχετικά με το σε πια πλατφόρμα μικροελεγκτή βρίσκεται συνδεδεμένος ο κάθε αίσθητήρας. Η αναζήτηση αποτελεσμάτων θα πρέπει να γίνεται βάση μιας τοποθεσίας που δίνει ο χρήστης και μιας ακτίνας έτσι ώστε να επιστρέφονται στο χρήστη μόνο οι αισθητήρες που βρίσκονται στη περιοχή γύρω από τη τοποθεσία που έχει επιλέξει. Επίσης, θα πρέπει να δίνεται η δυνατότητα τριών ειδών αναζήτησης για αποτελέσματα.

- Αναζήτηση για αισθητήρες: Θα παρουσιάζει σε μια λίστα όλους τους αισθητήρες που βρέθηκαν από την αναζήτηση ανεξάρτητα από τον τύπο τους και τη πλατφόρμα μικροελεγκτή στην οποία είναι συνδεδεμένοι.
- Αναζήτηση για πλατφόρμες μικροελεγκτών: Επιστρέφει στο χρήστη όλες τις πλατφόρμες μικροελεγκτών τις οποίες διαχειρίζεται μαζί με τους αισθητήρες που βρίσκονται συνδεδεμένοι στις πλατφόρμες αυτές και τις τελευταίες μετρήσεις του κάθε αισθητήρα. Δηλαδή, επιστρέφει τα ίδια αποτελέσματα με την αναζήτηση για αισθητήρες αλλά τα αποτελέσματα είναι ταξινομημένα ανά πλατφόρμα μικροελεγκτή.
- Αναζήτηση για τύπους αισθητήρων: Όπως την αναζήτηση για αισθητήρες αλλά φιλτράρει το αποτέλεσμα έτσι ώστε να επιστρέφονται οι τιμές ενός μόνο τύπου αισθητήρων π.χ. φωτός.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Απλός χρήστης, Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Προβολή ιστορικών μετρήσεων του κάθε αισθητήρα

Οι χρήστες του συστήματος θα πρέπει να είναι σε θέση να βλέπουν όλες τις προηγούμενες μετρήσεις του κάθε αισθητήρα. Αυτή η λειτουργία βοηθά στην εύρεση σημαντικών παρεκκλίσεων στις μετρήσεις και την ανίχνευση διαφόρων σφαλμάτων στους αισθητήρες. Επίσης, βοηθά στην αναγνώριση τιμών που ίσως υποδηλώνουν ότι κάτι δεν πάει καλά με το σύστημα υπό παρακολούθηση (π.χ. παραβίαση σπιτιού σε ένα σύστημα ασφαλείας). Θα υπάρχουν δύο τρόποι προβολής των ιστορικών τιμών του κάθε αισθητήρα:

  - Προβολή σε λίστα: Θα πρέπει να εμφανίζεται στο χρήστη μια λίστα με όλες τις προηγούμενες μετρήσεις του αισθητήρα που επιλέγεται ταξινομημένες σε αύξουσα σειρά με βάση τη χρονική στιγμή της μέτρησης της κάθε τιμής.
  - Προβολή σε γραφική παράσταση: Οι τιμές θα πρέπει να είναι διαθέσιμες και ως γραφική παράσταση παρουσιάζοντας τις τιμές στον άξονα των ψ και την χρονική στιγμή που λήφθηκε η κάθε μέτρηση στον άξονα των χ.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Απλός χρήστης, Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Αλλαγή βασικών ρυθμίσεων της εφαρμογής

Ο κάθε χρήστης θα πρέπει να μπορεί να αλλάξει τις βασικές ρυθμίσεις της εφαρμογής. Στις ρυθμίσεις αυτές περιλαμβάνονται το βασικό URL για πρόσβαση στις υπηρεσίες διαδικτύου, η ταυτότητα και ο όροφος του κτιρίου για το οποίο θέλει να βλέπει Anyplace Points Of Interest. Αυτό θα πρέπει να υπάρχει για να μπορούν οι χρήστες να κάνουν αναζήτηση και επεξεργασία των δικών τους δεδομένων που ίσως να είναι αποθηκευμένα σε κάποια άλλη υπηρεσία διαδικτύου. Οι ρυθμίσεις αυτές θα πρέπει να αλλάζουν κάθε φορά που ο χρήστης συνδέεται στην εφαρμογή και να δίνεται η δυνατότητα στο χρήστη να επιστρέψει στις προεπιλεγμένες ρυθμίσεις ανά πάσα στιγμή.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Απλός χρήστης, Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Ρύθμιση παραμέτρων συλλογής δεδομένων

Οι ιδιοκτήτες των αισθητήρων και οι διαχειριστές του συστήματος θα πρέπει να είναι σε θέση να αλλάζουν διάφορες παραμέτρους συλλογής δεδομένων από το σύστημα. Αυτές οι ρυθμίσεις αναλυτικά περιλαμβάνουν:

  - Απενεργοποίηση αισθητήρων: Οι ιδιοκτήτες των αισθητήρων θα πρέπει να μπορούν να απενεργοποιούν όλους τους αισθητήρες μέσω της εφαρμογής έτσι ώστε να μην λαμβάνουν πλέον μετρήσεις από αυτούς τους αισθητήρες.
  - Αλλαγή συχνότητας συλλογής δεδομένων από αισθητήρες: Για τον κάθε αισθητήρα, θα πρέπει να δίνεται η δυνατότητα αλλαγής της συχνότητας που συλλέγει και στέλνει μετρήσεις στο σύστημα. Οι συχνότητες αυτές θα ρυθμίζονται σε δευτερόλεπτα.
  - Αλλαγή συχνότητας συλλογής δεδομένων από τους μικροελεγκτές: Όπως και την αλλαγή της συχνότητας συλλογής δεδομένων από τους αισθητήρες αλλά θα ισχύει για όλους τους αισθητήρες που είναι συνδεδεμένοι σε μια πλατφόρμα μικροελεγκτή.
  - Απενεργοποίηση πλατφόρμας μικροελεγκτή: Όπως την απενεργοποίηση ενός αισθητήρα αλλά θα ισχύει για όλους τους αισθητήρες που είναι συνδεδεμένοι σε μια πλατφόρμα μικροελεγκτή.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Εισαγωγή νέων αισθητήρων στο σύστημα

Οι ιδιοκτήτες των αισθητήρων θα πρέπει να είναι σε θέση να εισάγουν νέους αισθητήρες στο σύστημα για παρακολούθηση των δεδομένων από αυτούς. Οι αισθητήρες που θα εισάγονται θα έχουν ένα συγκεκριμένο όνομα, τύπο (π.χ. θερμότητας) και θα πρέπει να δηλώνονται ως συνδεδεμένοι σε μια πλατφόρμα μικροελεγκτή. Στη συνέχεια ο υπεύθυνος για το νέο αισθητήρα θα διευθετεί την σύνδεσή του με ένα μικροελεγκτή και την διαδικασία εισαγωγής των μετρήσεών του στο σύστημα.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Ιδιοκτήτης αισθητήρων, Διαχειριστής.

- Εισαγωγή νέων μικροελεγκτών στο σύστημα

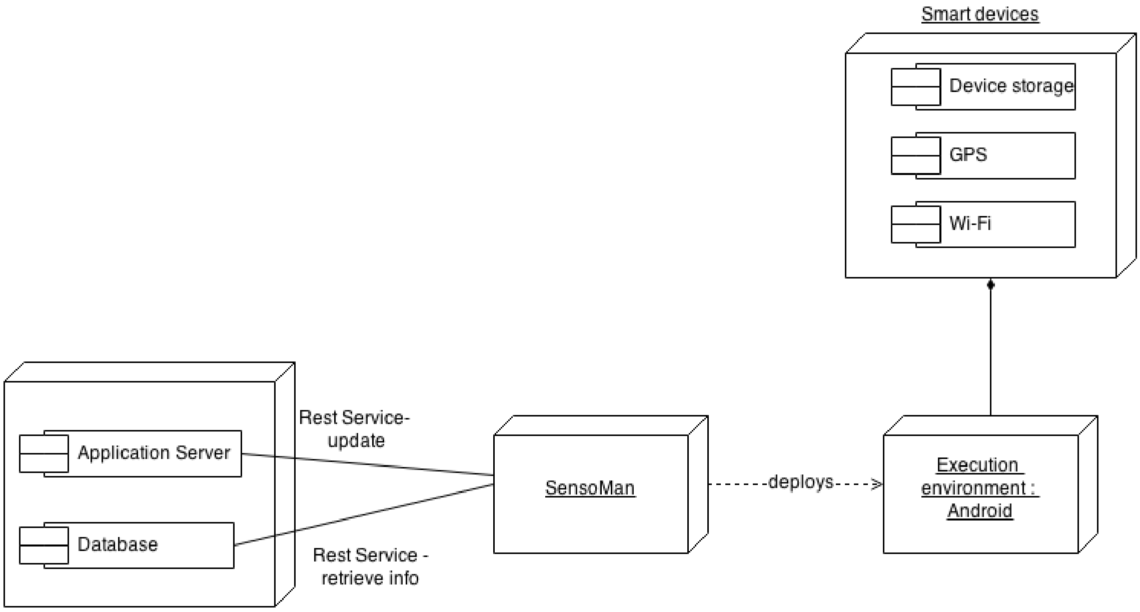
Οι ιδιοκτήτες αισθητήρων και οι διαχειριστές του συστήματος θα πρέπει να είναι σε θέση να εισάγουν νέους μικροελεγκτές στο σύστημα. Κατά τη διαδικασία εισαγωγής του μικροελεγκτή θα δίνεται ένα όνομα στον μικροελεγκτή καθώς και ο τύπος του, δηλαδή, αν είναι Raspberry Pi ή Arduino. Στη συνέχεια, ο υπεύθυνος για αυτό το μικροελεγκτή θα πρέπει να διευθετεί τη μέθοδο σύνδεσής του στο σύστημα, με άλλα λόγια τον τρόπο με τον οποίο ο μικροελεγκτής θα προσθέτει νέα δεδομένα για προβολή.

Ομάδες χρηστών που μπορούν να χρησιμοποιήσουν αυτή τη λειτουργία: Ιδιοκτήτης αισθητήρων, Διαχειριστής.

3.2 Σχεδιασμός συστήματος

Πιο κάτω εξηγείται αναλυτικά η σχεδίαση της εφαρμογής SensoMan και όλες οι κλάσεις που αποτελούν την εφαρμογή. Επίσης, γίνεται επεξήγηση της αλληλεπίδρασης της κινητής εφαρμογής με την υπηρεσία διαδικτύου. Υπενθυμίζεται ότι για εφαρμογές Android συνήθως κάθε κλάση με το όνομα activity έχει δική της οθόνη στην εφαρμογή. Η αναπαράσταση του σχεδιασμού του συστήματος γίνεται σε δύο μέρη. Το πρώτο μέρος περιγράφει πώς η κινητή εφαρμογή αλληλεπιδρά με την υπηρεσία διαδικτύου και το δεύτερο μέρος περιγράφει πώς είναι κτισμένη η εφαρμογή για κινητές συσκευές Android.

3.2.1 Αρχιτεκτονική συστήματος



Εικόνα 5: Αρχιτεκτονική σχεδίαση εφαρμογής

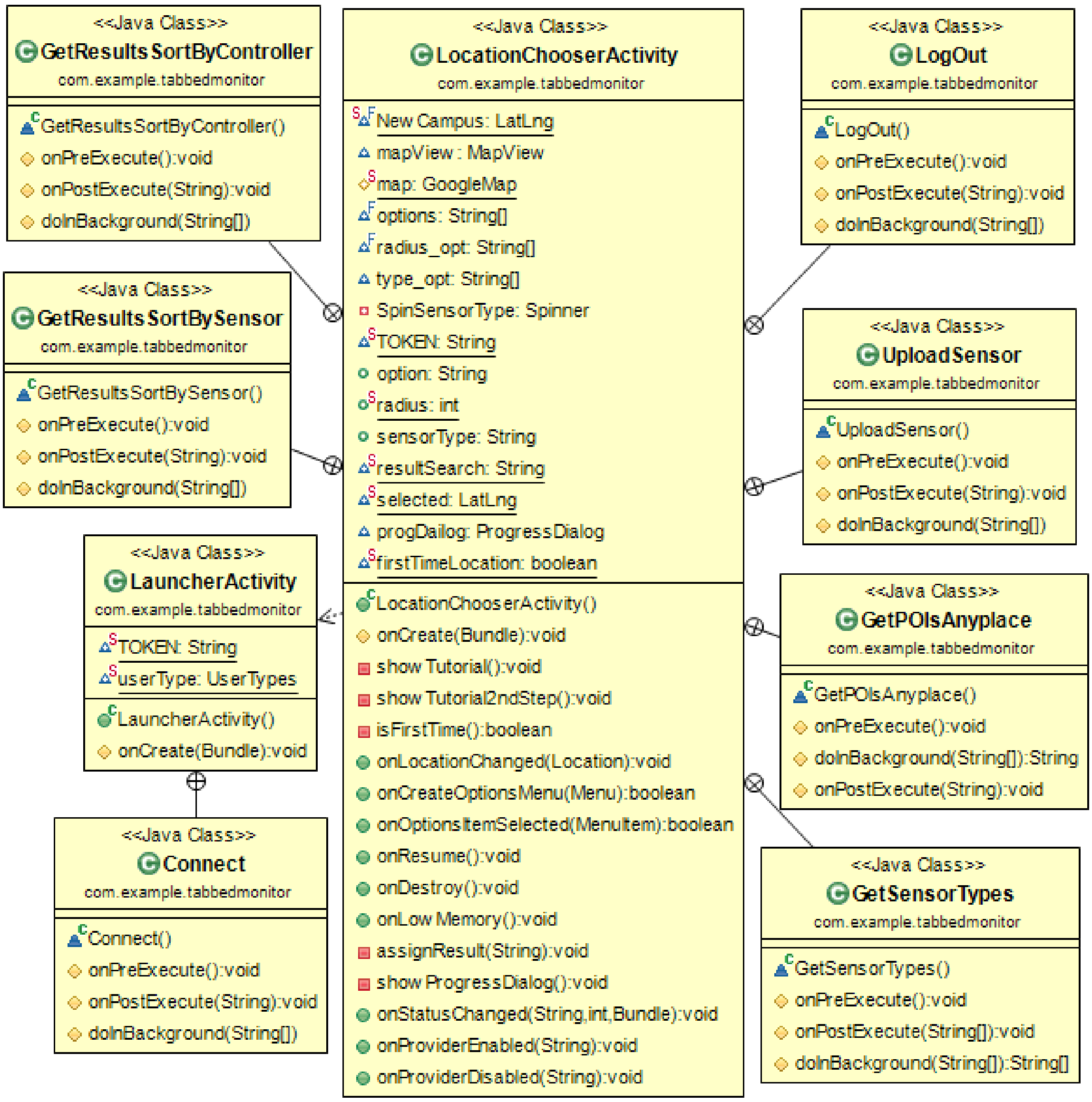
Στην πιο πάνω εικόνα φαίνεται η αρχιτεκτονική σχεδίαση της εφαρμογής και πώς αυτή χρησιμοποιεί συστατικά στοιχεία που παρέχει το περιβάλλον εκτέλεσής της. Επίσης, παρουσιάζεται ο τρόπος επικοινωνίας της εφαρμογής με την υπηρεσία διαδικτύου.

Συγκεκριμένα, η εφαρμογή αναπτύχθηκε για να τρέχει σε περιβάλλον εκτέλεσης Android με λειτουργικό σύστημα 4.1 Jellybean και πάνω. Αυτό επιτρέπει στην εφαρμογή να κάνει χρήση των αισθητήρων και των συστατικών στοιχείων που υπάρχουν σε κάθε κινητή συσκευή. Η εφαρμογή κάνει χρήση του GPS (global positioning system) που είναι διαθέσιμο σε κάθε κινητή συσκευή για ανεύρεση και προβολή της τοποθεσίας του χρήστη. Ακόμα μία σημαντική λειτουργία που παρέχουν οι κινητές συσκευές είναι η ασύρματη πρόσβαση στο διαδίκτυο. Η λειτουργία αυτή χρησιμοποιείται εκτενώς από την εφαρμογή αφού χρησιμεύει στην επικοινωνία με την υπηρεσία διαδικτύου η οποία διαθέτει όλες τις απαραίτητες πληροφορίες που χρειάζεται η εφαρμογή. Τέλος, η εφαρμογή κάνει χρήση του αποθηκευτικού χώρου που υπάρχει στην κινητή συσκευή για αποθήκευση κάποιων πληροφοριών που έχουν να κάνουν με την εκτέλεση της.

Από την άλλη μεριά, όλες οι υπόλοιπες λειτουργίες της εφαρμογής γίνονται με εκτεταμένη χρήση της υπηρεσίας διαδικτύου. Η υπηρεσία διαδικτύου αποτελείται από ένα εξυπηρετητή εφαρμογών και μια βάση δεδομένων. Ο εξυπηρετητής εφαρμογών είναι υπεύθυνος να ακούει για αιτήσεις από εφαρμογές και να απαντά με τα κατάλληλα δεδομένα ή να εκτελεί τις κατάλληλες λειτουργίες για επεξεργασία των δεδομένων. Η βάση δεδομένων του συστήματος χρησιμεύει στην αποθήκευση όλων των δεδομένων που έχουν να κάνουν με τους αισθητήρες δηλαδή όλες τις μετρήσεις τους και πληροφορίες σχετικά με τις πλατφόρμες μικροελεγκτών στις οποίες είναι συνδεδεμένοι, τον τύπο τους και την τοποθεσία τους.

Κάθε φορά που ο χρήστης της κινητής εφαρμογής κάνει αίτηση για λήψη δεδομένων, η εφαρμογή αποστέλλει ένα αίτημα στην υπηρεσία διαδικτύου. Στη συνέχεια, ο εξυπηρετητής εφαρμογών αναλαμβάνει την εξυπηρέτηση του αιτήματος με την ανάκτηση των σχετικών δεδομένων από τη βάση δεδομένων και αποστολή τους στην εφαρμογή σε μια προκαθορισμένη μορφή. Στη συνέχεια, η εφαρμογή είναι υπεύθυνη για την επεξεργασία της απάντησης από τον εξυπηρετητή εφαρμογών και προβολή των δεδομένων σε μορφή εύκολα κατανοητή από το χρήστη.

3.2.2 UML class diagram



Εικόνα 6:SensoMan UML diagram 1

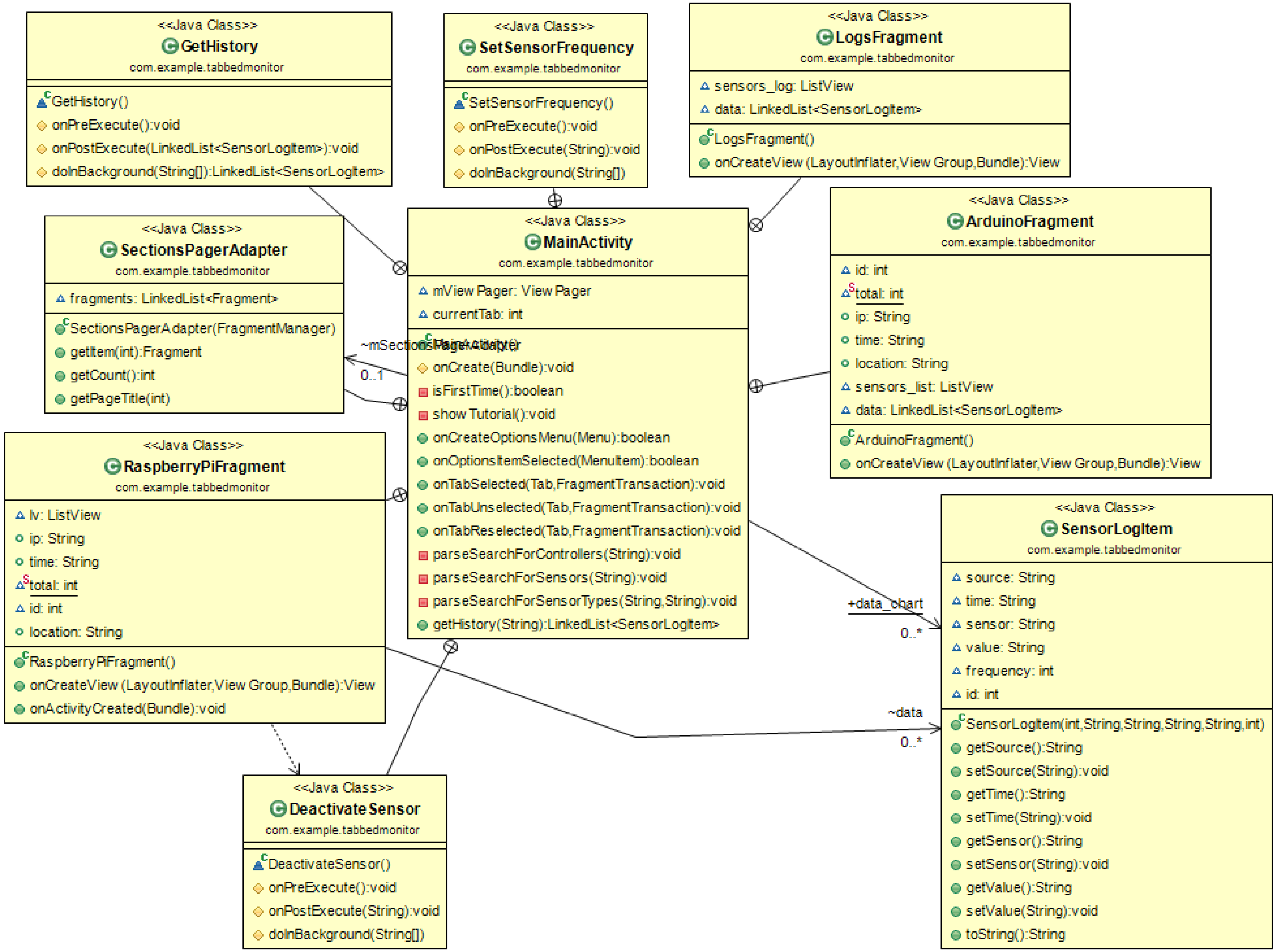
Σε αυτό το διάγραμμα φαίνεται η πρώτη κύρια κλάση του συστήματος η οποία ονομάζεται LocationChooserActivity. Αυτή η κλάση αποτελεί την πρώτη οθόνη που βλέπει ο χρήστης όταν συνδέεται στην εφαρμογή και είναι προσβάσιμη από την κλάση LauncherActivity που χρησιμεύει για τη σύνδεση του χρήστη.

LauncherActivity: Αυτή η κλάση εμφανίζει στο χρήστη πεδία για εισαγωγή των στοιχείων του έτσι ώστε να συνδεθεί στο σύστημα. Στη συνέχεια καλείται η κλάση Connect για έλεγχο των στοιχείων και ανάθεση του token που θα χρησιμοποιηθεί σε όλη την εφαρμογή. Επίσης, ανατίθεται ο τύπος χρήστη της εφαρμογής για καθορισμό των ενεργειών που θα είναι διαθέσιμες στο χρήστη καθ' όλη τη διάρκεια χρήσης της εφαρμογής. Μόλις γίνει η επικύρωση των στοιχείων του χρήστη, καλείται η πρώτη κύρια οθόνη του συστήματος που αντιπροσωπεύεται από την κλάση LocationChooserActivity. Αυτή η κλάση παρουσιάζει στο χρήστη την οθόνη με το χάρτη, έτσι ώστε να μπορεί να επιλέξει τοποθεσία και τις παραμέτρους αναζήτησης. Αυτή η κλάση περιέχει ένα μεγάλο αριθμό υπό-κλάσεων που υλοποιούν την επικοινωνία με την υπηρεσία διαδικτύου. Η κλάση αυτή περιέχει πεδία που αποθηκεύουν το χάρτη, τις παραμέτρους αναζήτησης, τον τύπο του χρήστη που χρησιμοποιεί την εφαρμογή και άλλα πεδία δεδομένων που χρησιμεύουν στη λειτουργία της εφαρμογής. Επίσης, περιέχει μεθόδους που καλούνται σε διάφορες ενέργειες του χρήστη αλλά και μεθόδους που χρησιμεύουν στην αναζήτηση όπως τον εντοπισμό της τοποθεσίας του χρήστη.

Οι υπό-κλάσεις που υλοποιεί η συγκεκριμένη κλάση έχουν ως εξής:

- GetResultsSortByController: Η κλάση αυτή υλοποιεί τη λειτουργία αναζήτησης με βάση τις πλατφόρμες μικροελεγκτών. Συγκεκριμένα, εκτελεί ένα HTTP request στην υπηρεσία διαδικτύου και παίρνει τα αποτελέσματα σε μορφή JSON.
- GetResultsSortBySensor: Η κλάση αυτή υλοποιεί τη λειτουργία αναζήτησης με βάση τους αισθητήρες. Συγκεκριμένα, εκτελεί ένα HTTP request στην υπηρεσία διαδικτύου και παίρνει τα αποτελέσματα σε μορφή JSON.
- GetSensorTypes: Η κλάση αυτή χρησιμεύει στη λήψη όλων των τύπων αισθητήρων που βρίσκονται συνδεδεμένοι σε κάποιο μικροελεγκτή. Συγκεκριμένα, εκτελεί ένα HTTP request στην υπηρεσία διαδικτύου και παίρνει τα αποτελέσματα σε μορφή JSON.
- LogOut: Η κλάση αυτή χρησιμεύει στην αποσύνδεση ενός χρήστη από την εφαρμογή
- UploadSensor: Η κλάση αυτή υλοποιεί τη λειτουργία προσθήκης ενός νέου αισθητήρα στο σύστημα. Συγκεκριμένα, εκτελεί ένα HTTP request στην υπηρεσία διαδικτύου στέλνοντας το όνομα του αισθητήρα που θα προστεθεί, τον τύπο του και το μικροελεγκτή στον οποίο είναι συνδεδεμένος ο συγκεκριμένος αισθητήρας.

- GetPOIsAnyplace: Η κλάση αυτή χρησιμεύει στην ανάκτηση σημείων ενδιαφέροντος που προστέθηκαν προηγουμένως στην υπηρεσία Anyplace. Αυτό γίνεται με HTTP request στην REST υπηρεσία διαδικτύου που προσφέρει το Anyplace και παίρνοντας τα αποτελέσματα σε μορφή JSON. Στη συνέχεια, εμφανίζει τα αποτελέσματα στο χάρτη που φαίνεται στην οθόνη της κινητής συσκευής.



Εικόνα 7: SensoMan UML diagram 2

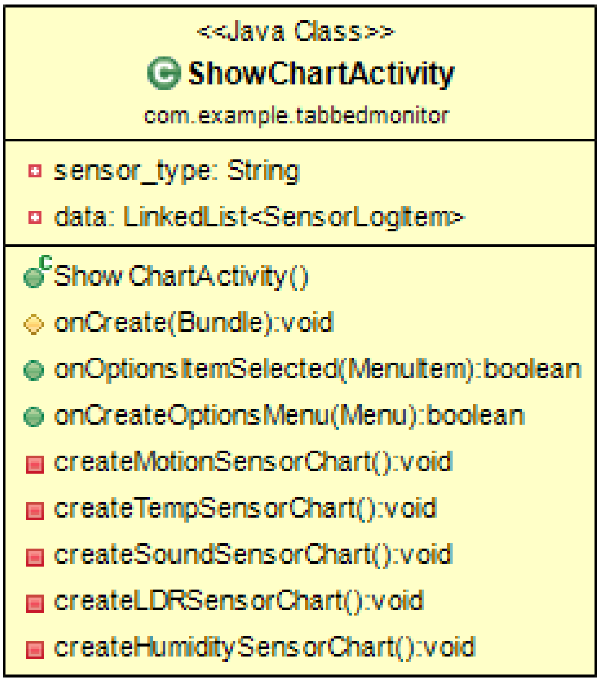
Στο δεύτερο μέρος του διαγράμματος UML φαίνεται η δεύτερη κύρια κλάση της εφαρμογής που αποτελεί και την πιο κύρια οθόνη της εφαρμογής με το όνομα MainActivity. Συγκεκριμένα, στη διεπαφή αυτής της κλάσης μπαίνει ένας αριθμός από ArduinoFragment, RasperryPiFragment ή SensorLogFragment με τη βοήθεια μιας ειδικής κλάσης με το όνομα SectionsPageAdapter. Από την κύρια κλάση που φαίνεται πιο πάνω υλοποιούνται μέθοδοι και κλάσεις που χρησιμεύουν στην επεξεργασία των δεδομένων που έχουν ληφθεί από την υπηρεσία διαδικτύου και στην ανάκτηση περισσότερων πληροφοριών από την υπηρεσία διαδικτύου. Οι υπό-κλάσεις αυτής της κλάσης εξηγούνται αναλυτικότερα πιο κάτω. Η κλάση MainActivity παρέχει επίσης μεθόδους που βοηθούν στη διεκπεραίωση κάποιων λειτουργιών όπως την προβολή ενός μικρού εγχειριδίου χρήσης κατά την είσοδο του χρήστη στην εφαρμογή.

Κατ' αρχάς η κύρια κλάση παρέχει τρείς υπό-κλάσεις που χρησιμεύουν στη διαφορετική αναπαράσταση των διαφόρων αποτελεσμάτων. Συγκεκριμένα, οι κλάσεις ArduinoFragment, RaspberryPiFragment και SensorLogFragment χρησιμεύουν στη προβολή των δεδομένων από το χρήστη. Οι τρείς αυτές κλάσεις έχουν σχεδόν την ίδια λειτουργικότητα αφού και οι τρείς ενσωματώνουν μια λίστα από αντικείμενα τύπου SensorLogItem που αναπαριστούν τα δεδομένα. Τα πεδία των κλάσεων αυτών χρησιμεύουν στην αναπαράσταση τους ως μικροελεγκτές και περιλαμβάνουν την τοποθεσία του κάθε μικροελεγκτή, την ταυτότητά του και την διεύθυνση ip του που βρίσκεται αποθηκευμένη στη βάση δεδομένων της υπηρεσίας διαδικτύου. Επίσης, διαθέτουν μεθόδους επεξεργασίας αυτών των δεδομένων καλώντας άλλες κλάσεις που αναλαμβάνουν την αλλαγή της συχνότητας των αισθητήρων και την απενεργοποίηση των αισθητήρων.

- DeactivateSensor: Η κλάση αυτή χρησιμεύει στην απενεργοποίηση ενός αισθητήρα κάνοντας κλήση στην υπηρεσία διαδικτύου και στέλνοντας τον αριθμό του αισθητήρα που θα απενεργοποιηθεί.
- GetHistory: Η κλάση αυτή χρησιμεύει στην ανάκτηση ιστορικών τιμών από τους αισθητήρες. Συγκεκριμένα, το επιτυγχάνει αυτό με μια κλήση στην υπηρεσία διαδικτύου με τον αριθμό του αισθητήρα και επεξεργάζεται το αποτέλεσμα που επιστρέφεται υπό μορφή JSON τοποθετώντας το σε μια λίστα.

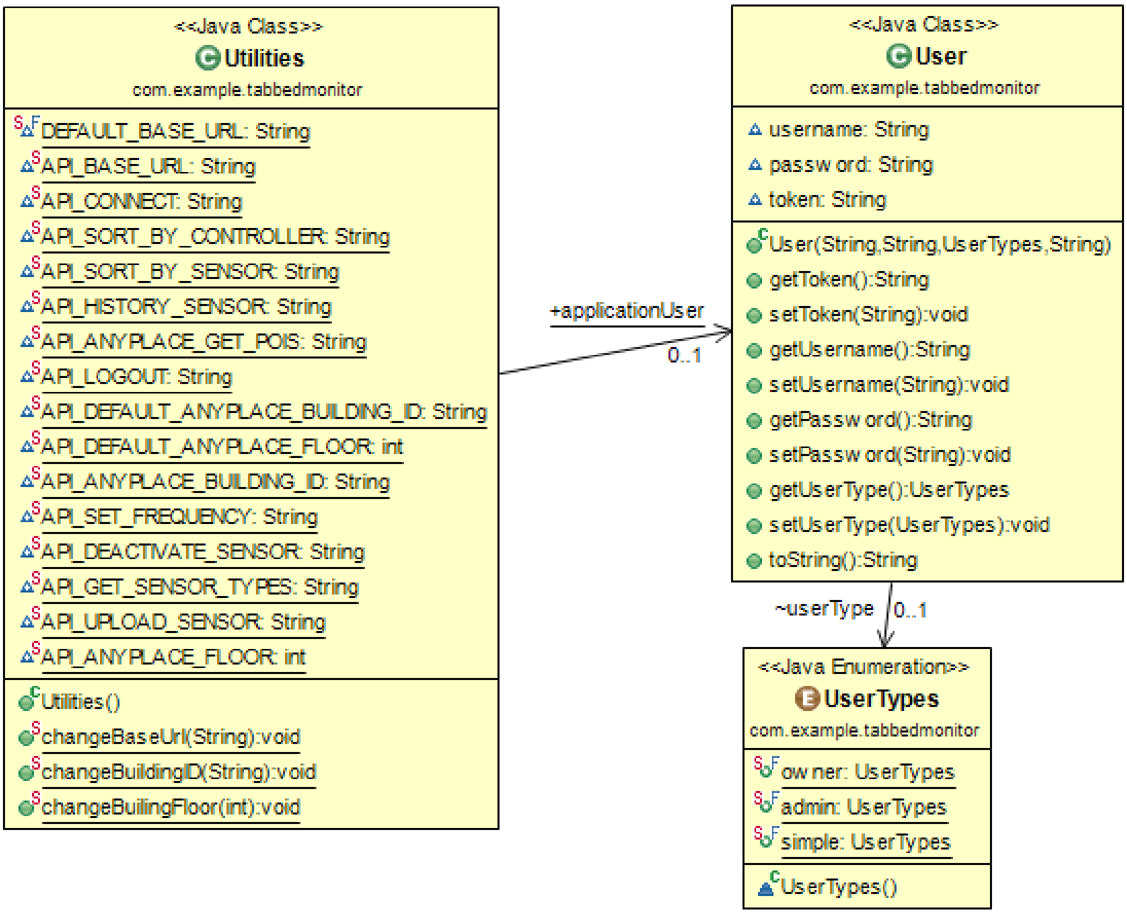
- SetSensorFrequency: Η κλάση αυτή χρησιμοποιείται στην περίπτωση που ο χρήστης αλλάζει την συχνότητα κάποιου αισθητήρα. Κατά τη κλήση αυτής της κλάσης γίνεται μια κλήση στην υπηρεσία διαδικτύου με τον αριθμό του αισθητήρα και την νέα συχνότητα που καθορίζεται από το χρήστη.

Τέλος, η κλάση SensorLogItem αποτελεί την αναπαράσταση του αποτελέσματος που επιστρέφεται από την υπηρεσία διαδικτύου σε ένα αντικείμενο. Τα πεδία source,time,sensor,value,frequency,id αποτελούν χαρακτηριστικά στοιχεία της κάθε μέτρησης που υπάρχει στη βάση δεδομένων και αναπαριστούν το μικροελεγκτή, ώρα, όνομα, τιμή, συχνότητα και αριθμό του κάθε αισθητήρα αντίστοιχα.
Η κλάση SectionsPageAdapter χρησιμεύει στη διάταξη των στοιχείων διεπαφής στην οθόνη και δεν σχετίζεται με τη λειτουργικότητα του συστήματος.



Εικόνα 8: SensoMan UML diagram 3

Η κλάση αυτή αποτελεί την τρίτη οθόνη της διεπαφής χρήστη και χρησιμοποιείται αποκλειστικά για τι προβολή των δεδομένων ενός αισθητήρα σε μορφή γραφικής παράστασης. Τα πεδία της κλάσης αποτελούν μια λίστα από μετρήσεις και ο τύπος του αισθητήρα. Η κλάση αυτή διαθέτει μεθόδους για δημιουργία γραφικών παραστάσεων ανάλογα με τους διαφορετικούς τύπους αισθητήρων.



Εικόνα 9: SensoMan UML diagram 4

Τέλος, οι τελευταίες κλάσεις που αποτελούν το σύστημα είναι οι κλάσεις Utilities και User. Αυτές οι δύο κλάσεις είναι βοηθητικές για τη λειτουργία της εφαρμογής.

- Utilities: Αυτή η κλάση περιέχει όλα τα απαραίτητα URLs για κλήση των υπηρεσιών διαδικτύου και πεδία που αφορούν ρυθμίσεις της εφαρμογής. Επίσης, δίνει κάποιες μεθόδους για αλλαγή των ρυθμίσεων της εφαρμογής. Τέλος, σε αυτή τη κλάση αποθηκεύεται ο ενεργός χρήστης της εφαρμογής.

- User: Η κλάση αυτή αναπαριστά ένα χρήστη του συστήματος. Περιέχει πεδία που αναπαριστούν το username, password και το token για το συγκεκριμένο χρήστη. Επίσης, περιέχει ένα πεδίο που αναπαριστά τον τύπο του χρήστη που είναι συνδεδεμένος στην εφαρμογή.

4 Τεχνολογίες Κινητής Εφαρμογής

4.1 Ανάπτυξη εφαρμογών Android

Η ανάπτυξη εφαρμογών Android αναφέρεται στη δημιουργία λογισμικού για συσκευές με λειτουργικό σύστημα Android. Συνήθως, η ανάπτυξη τέτοιου είδους εφαρμογών γίνεται σε Java όμως χρησιμοποιούν το Android SDK το οποίο προορίζεται ειδικά για Android εφαρμογές. Το Android SDK παρέχει APIs που χρησιμεύουν στην υλοποίηση εφαρμογών που εκμεταλλεύονται αισθητήρες εισόδους/εξόδους που υπάρχουν σε κινητές συσκευές όπως το accelerometer που χρησιμεύει στην ανίχνευση και περιγραφή της κίνησης της συσκευής ή το GPS και το Wi-Fi.

Οι εφαρμογές Android είναι κτισμένες σαν ένα σύνολο από ξεχωριστά συστατικά στοιχεία που μπορούν να κληθούν ανεξάρτητα το ένα από το άλλο. Για παράδειγμα, κάθε ξεχωριστό activity παρέχει μια μοναδική οθόνη για αλληλεπίδραση με το χρήστη ενώ ένα service τρέχει ξεχωριστά από μόνο του στο παρασκήνιο.

Οι εφαρμογές Android έχουν στο σκελετό τους μια συλλογή από activities. Το κάθε activity μπορεί να περιγραφεί ως μια ξεχωριστή κλάση στην υλοποίηση. Όπως προαναφέρθηκε, κάθε activity παρέχει μια ξεχωριστή οθόνη για αλληλεπίδραση με το χρήστη. Το Android αναλαμβάνει να ακούει για χειρονομίες του χρήστη όπως touch events και να τα μεταφράζει σε κλήσεις μεθόδων όπως onClickListener( ) που υλοποιούν κάποια λειτουργία. Ο σχεδιασμός της διεπαφής χρήστη γίνεται σε αρχεία

XML που καθορίζουν την παράταξη των διαφόρων στοιχείων όπως κουμπιά, λίστες, πεδία εισόδου δεδομένων, εικόνες κλπ.

Το μοντέλο υλοποίησης εφαρμογών Android παρέχει τη δυνατότητα πολλών σημείων εισόδου σε ένα activity. Για παράδειγμα, από ένα συστατικό στοιχείο μπορεί να κληθεί ένα άλλο συστατικό στοιχείο με τη χρήση μιας ειδικής κλάσης που ονομάζεται intent. Το intent δηλώνει μια πρόθεση για ενεργοποίηση ενός συστατικού στοιχείου που μπορεί να βρίσκεται είτε στην ίδια εφαρμογή είτε σε ξεχωριστή εφαρμογή. Μέσω του intent επίσης μπορούν δύο συστατικά στοιχεία να επικοινωνήσουν προσθέτοντας σημαντικές πληροφορίες μέσα στο αντικείμενο intent έτσι ώστε να είναι διαθέσιμα στο το άλλο συστατικό στοιχείο.

Αυτό το μοντέλο, όπως προαναφέρθηκε επιτρέπει πολλά σημεία πρόσβασης σε μια εφαρμογή αλλά επίσης επιτρέπει σε μια εφαρμογή να λειτουργεί ως τη «προκαθορισμένη» επιλογή για οποιαδήποτε λειτουργία που μπορεί να καλεί πάνω από μια εφαρμογές (π.χ. το άνοιγμα μιας ιστοσελίδας).

Σημειώνεται ότι για πρόσβαση σε συγκεκριμένους αισθητήρες, συστατικά στοιχεία ή πληροφορίες της συσκευής χρειάζεται ένας ειδικός τύπος άδειας που ονομάζεται permission.
Κάθε εφαρμογή Android που θέλει να χρησιμοποιήσει κάποιο συστατικό στοιχείο που χρειάζεται άδεια θα πρέπει να δηλώνει ότι χρειάζεται το συγκεκριμένο permission στο αρχείο που περιγράφει την εφαρμογή (AndroidManifest.xml).

Τέλος, οι εφαρμογές Android μπορούν να εμπλουτιστούν με βιβλιοθήκες, είτε αυτές προορίζονται γενικά για Java κώδικα είτε βιβλιοθήκες χτισμένες συγκεκριμένα για το Android που εκμεταλλεύονται τις δυνατότητες που προσφέρει το Android.

Περισσότερες πληροφορίες για το Android SDK, Android APIs, design, activities κλπ. Υπάρχουν στη βιβλιογραφία [4].

Στο επόμενο υποκεφάλαιο θα γίνει αναφορά σε κάποιες από τις βιβλιοθήκες που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής SensoMan.

4.2 Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση

Κατά την υλοποίηση της εφαρμογής κρίθηκε απαραίτητη η χρήση βιβλιοθηκών για την υλοποίηση κάποιων λειτουργιών που δεν παρέχονταν άμεσα από το Android SDK. Συγκεκριμένα, χρησιμοποιήθηκαν κάποιες βιβλιοθήκες για τη δημιουργία γραφικών παραστάσεων με τις μετρήσεις από τους αισθητήρες και για τη δημιουργία ενός οδηγού χρήσης της εφαρμογής μέσα στην ίδια την εφαρμογή. Αυτές οι βιβλιοθήκες και η λειτουργία τους περιγράφονται πιο κάτω.

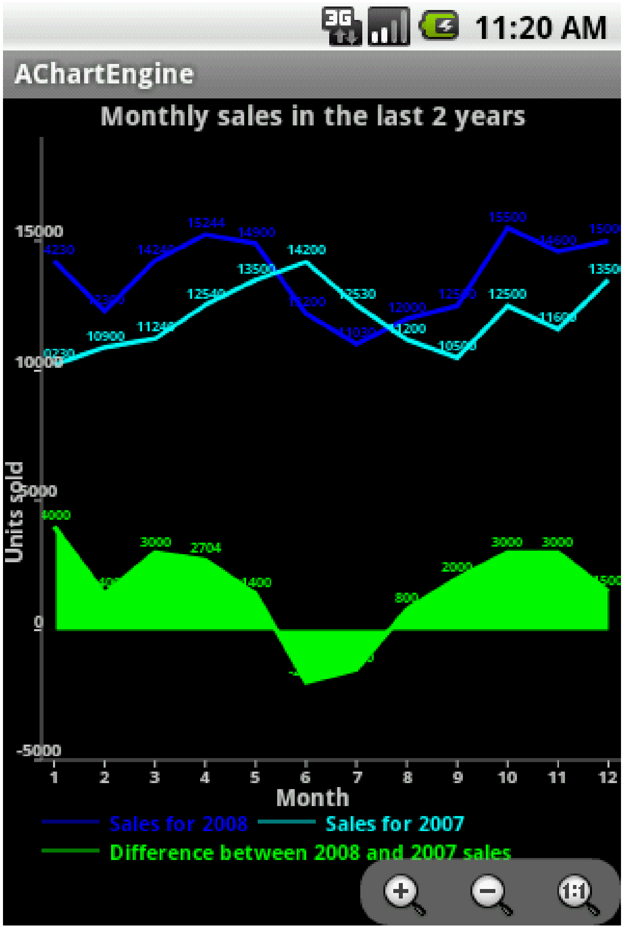
4.2.1 Achartengine – Γραφικές παραστάσεις

Η βιβλιοθήκη aChartEngine [13] χρησιμοποιείται για δημιουργία γραφικών παραστάσεων στο Android. Μέχρι τώρα υποστηρίζει τους πιο κάτω τύπους γραφικών παραστάσεων:

- line chart
- area chart
- scatter chart
- time chart
- bar chart
- pie chart
- bubble chart
- doughnut chart
- range (high-low) bar chart
- dial chart / gauge

- combined (any combination of line, cubic line, scatter, bar, range bar, bubble) chart
- cubic line chart

Η βιβλιοθήκη κυκλοφορεί κάτω από την άδεια Apache License 2.0 και παρέχει αρκετές δυνατότητες για τη δημιουργία γραφικών παραστάσεων όπως πολλαπλές σειρές δεδομένων και μορφοποίησης του τρόπου προβολής των δεδομένων.

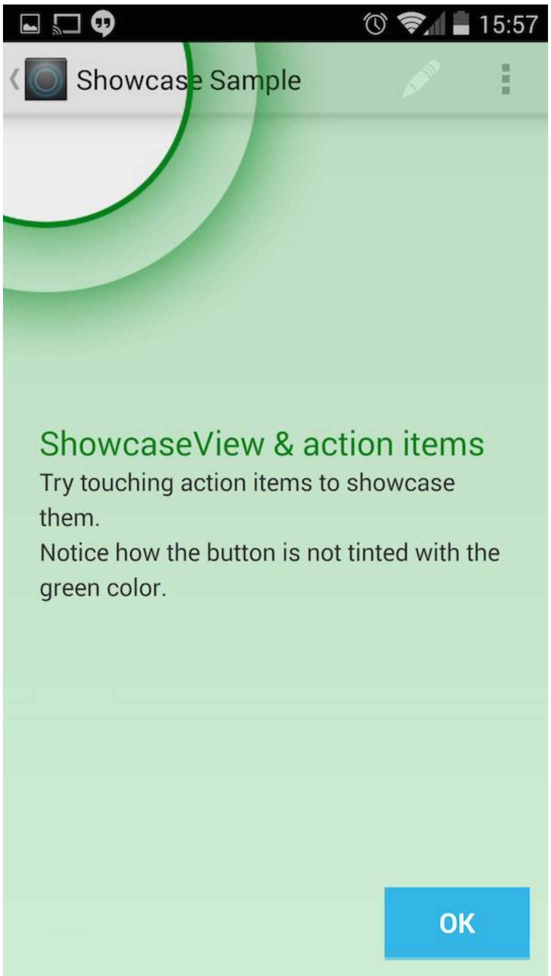


Εικόνα 10: Παράδειγμα χρήσης της βιβλιοθήκης achartengine

Σημειώνεται ότι υπάρχει ένας μεγάλος αριθμός βιβλιοθηκών που υποστηρίζουν τη δημιουργία γραφικών παραστάσεων στο Android αλλά έχει χρησιμοποιηθεί η συγκεκριμένη βιβλιοθήκη λόγω της απλότητας της και ευκολίας στη χρήση της.

4.2.2 ShowcaseView – Δημιουργία on-screen tutorials

Η βιβλιοθήκη ShowcaseView [14] χρησιμοποιείται για τη ανάδειξη συγκεκριμένων στοιχείων της διεπαφής χρήστη με ένα ελκυστικό τρόπο για δημιουργία οδηγιών χρήσης των συστατικών στοιχείων της οθόνης. Για τους σκοπούς της εφαρμογής χρησιμοποιήθηκε η συγκεκριμένη βιβλιοθήκη για προβολή ενός tutorial κατά την εκκίνηση της εφαρμογής μετά την εγκατάστασή της που θα εξηγεί στους χρήστες τη χρήση της εφαρμογής. Η βιβλιοθήκη αυτή κυκλοφορεί κάτω από την άδεια Apache License 2.0.



Εικόνα 11: Παράδειγμα χρήσης της βιβλιοθήκης ShowcaseView

4.3 RESTful Services

Για την ανάκτηση των δεδομένων και τη αποστολή πληροφοριών στην υπηρεσία διαδικτύου χρησιμοποιήθηκαν τα πιο κάτω REST services API calls:

4.3.1 Σύνδεση με την υπηρεσία διαδικτύου

Connect to RESTful Service

Description: Connect to the RESTful Service using credentials (**username + password**) as POST parameters to the HTTP Request. A valid token is returned to the response, so the user will send token in every request after connection.
URL = http://thesis.in.cs.ucy.ac.cy/datasensors/connect.php
Response Example: Successfully connected and a valid token is received.


```
{
   "status": "success",
   "request time": "2015-02-24 06:02:25",
   "IP Address": "10.16.19.32",
   "message": "Successfully connected.",
   "token": "************************",
   "userType": "owner"
}
```


Εικόνα 12: Connect to REST service API call

Η πιο πάνω κλήση χρησιμεύει στην σύνδεση με την υπηρεσία διαδικτύου. Συγκεκριμένα αποστέλλονται σαν POST παράμετροι username και το password του χρήστη της εφαρμογής και η υπηρεσία διαδικτύου επιστρέφει πίσω ένα token το οποίο χρησιμοποιείται καθ’ όλη τη διάρκεια χρήσης της εφαρμογής και αποστέλλεται μαζί με όλες τις υπόλοιπες κλήσεις στην υπηρεσία διαδικτύου.

4.3.2 Ανάκτηση Δεδομένων

Οι πιο κάτω κλήσεις χρησιμοποιούνται κατά τη διαδικασία ανάκτησης δεδομένων μέσω της υπηρεσίας διαδικτύου.

Base URL = http://thesis.in.cs.ucy.ac.cy/datasensors/retrieve/

4.3.2.1 Αναζήτηση με βάση μικροελεγκτές

Active micro-controllers

Description: Get the active micro-controllers info within a distance and their active connected sensors info with their last measurement. The valid token (**token**), **latitude**, **longitude** and **radius** in KM are given as POST parameters.
URL: Base URL +micro-controllers.php
Response Example: Successfully retrieved 1 or more micro-controllers


```
{
   "status": "success",
   "request time": "2015-02-24 06:02:57",
   "IP Address": "10.16.19.32",
   "Location Requested": {
      "latitude": "35.144343",
      "longitude": "33.411382",
      "radius": "5KM"
   },
   "message": "Successfully retrieved.",
   "count": 2,
   "controllers": [
      {
         "Controller ID": "2",
         "Controller Name": "raspberryPi-1",
         "Sensors": {
            "status": "success",
            "message": "Successfully retrieved",
            "count": 0,
            "resultSensors": []
         }
      },
      {
         "Controller ID": "1",
         "Controller Name": "arduino-1",
         "Sensors": {
            "status": "success",
            "message": "Successfully retrieved",
            "count": 5,
            "resultSensors": [
               {
                  "Sensor ID": "1",
                  "Sensor Name": "11304P562341b",
                  "Sensor Type": "humidity",
                  "SensorFrequency": "3",
                  "Measurement ID": "11",
```


Εικόνα 13: REST service retrieve data API call

Η πιο πάνω κλήση χρησιμοποιείται κατά την ανάκτηση των δεδομένων που βρίσκονται στη βάση δεδομένων της υπηρεσίας διαδικτύου. Συγκεκριμένα, αποδέχεται ως παραμέτρους το token, την τοποθεσία που επέλεξε ο χρήστης και την ακτίνα σε χιλιόμετρα που επέλεξε ο χρήστης ως βάση της αναζήτησης. Στη συνέχεια, επιστρέφει ως λίστα όλους τους μικροελεγκτές που βρίσκονται στη συγκεκριμένη ακτίνα γύρω από τη τοποθεσία που επέλεξε ο χρήστης μαζί με τις τελευταίες μετρήσεις που πήραν αυτοί οι αισθητήρες. Αυτή η κλήση χρησιμοποιείται για την αναζήτηση με βάση τους μικροελεγκτές.

Οι κλήσεις για την ανάκτηση δεδομένων βάση αισθητήρων και του ιστορικού του κάθε αισθητήρα λειτουργούν παρόμοια και για αυτό το λόγο δε θα εξηγηθούν σε αυτή την ενότητα.
Το ίδιο ισχύει και για την κλήση για ανάκτηση όλων των τύπων των αισθητήρων που βρίσκονται ενωμένοι με το σύστημα. Οι κλήσεις των πιο πάνω αναφέρονται πιο κάτω.

4.3.2.2 Αναζήτηση με βάση αισθητήρες

Description: Get the active sensors info within a distance and their last measurement. The valid token (token), latitude, longitude and radius in KM are given as POST parameters.
URL: Base URL +sensors.php

4.3.2.3 Ανάκτηση ιστορικού αισθητήρα

Description: Get all the measurements of the sensor given as parameter. The valid token and the sensorID (int) are given as POST parameters.
URL: Base URL + historySensor.php

4.3.2.4 Ανάκτηση όλων των τύπων αισθητήρων

Description: Get all the types of sensors. The valid token is given as a POST parameter.
URL: Base URL +sensor_properties.php

4.3.3 Αποστολή δεδομένων

Οι πιο κάτω κλήσεις χρησιμοποιούνται στην αποστολή δεδομένων προς την υπηρεσία διαδικτύου

Base URL = http://thesis.in.cs.ucy.ac.cy/datasensors/upload/

4.3.3.1 Απενεργοποίηση αισθητήρα

Deactivate Sensor

Description: Deactivate a sensor. The valid **token** and the **sensorID** (int) are given as POST parameters.
URL: Base URL + deactivate_sensor.php
Response Example: Fail to deactivate sensor because is already deactivated.


```
{
    "status": "fail",
    "request time": "2015-02-24 06:02:03",
    "IP Address": "10.16.19.32",
    "message": "The sensor is already deactivated"
}
```


Εικόνα 14: Απενεργοποίηση αισθητήρα REST API call

Η πιο πάνω κλήση χρησιμοποιείται για την απενεργοποίηση ενός αισθητήρα. Συγκεκριμένα λαμβάνει ως παραμέτρους τη ταυτότητα ενός αισθητήρα και το token που δημιουργήθηκε κατά τη σύνδεση στην εφαρμογή και επιστρέφει το κατά πόσο ήταν επιτυχημένη η απενεργοποίηση του συγκεκριμένου αισθητήρα.

4.3.3.2 Αλλαγή συχνότητας αισθητήρα

Η κλήση για την αλλαγή συχνότητας αισθητήρα λειτουργεί με παρόμοιο τρόπο με τη κλήση για απενεργοποίηση του αισθητήρα και για αυτό δε θα εξηγηθεί σε αυτή την ενότητα.

<u>Description</u>: Change the frequency of the sensor (collects data). The valid token, the sensorID (int) and the frequency (int) are given as POST parameters.

<u>URL</u>: Base URL +frequency.php

4.3.3.3 Εισαγωγή νέου αισθητήρα στο σύστημα

Upload Sensor

Description: Upload a sensor. The valid **token**, the **sensorName** (string) and the **type** of sensor (int) are given as POST parameters.

URL: Base URL + upload_sensor.php

Response Example: Successfully add a sensor.


```
{
   "status": "success",
   "request time": "2015-02-24 06:02:52",
   "IP Address": "10.16.19.32",
   "message": "Successfully added"
}
```


Εικόνα 15: Εισαγωγή νέου αισθητήρα REST service API call

Η πιο πάνω κλήση χρησιμεύει στην εισαγωγή ενός νέου αισθητήρα στο σύστημα. Συγκεκριμένα αποστέλλονται ως παράμετροι ο τύπος του αισθητήρα, το όνομα του και το token που δημιουργήθηκε κατά την σύνδεση στην υπηρεσία διαδικτύου. Στη συνέχεια, η υπηρεσία απαντά με το κατά πόσο ήταν επιτυχημένη η εισαγωγή του αισθητήρα στο σύστημα.

4.3.4 Περισσότερες πληροφορίες για την υπηρεσία διαδικτύου

Η υπηρεσία διαδικτύου που περιγράφηκε πιο πάνω αποτελεί μέρος της ατομικής διπλωματικής εργασίας της φοιτήτριας Στέλλας Κωνσταντίνου. Για περισσότερες πληροφορίες σχετικά με τις πιο πάνω κλήσεις αλλά και για όλο το API της υπηρεσίας διαδικτύου παρακαλώ όπως προτρέξετε στην ατομική διπλωματική εργασία της Στέλλας Κωνσταντίνου **Error! Reference source not found.**.

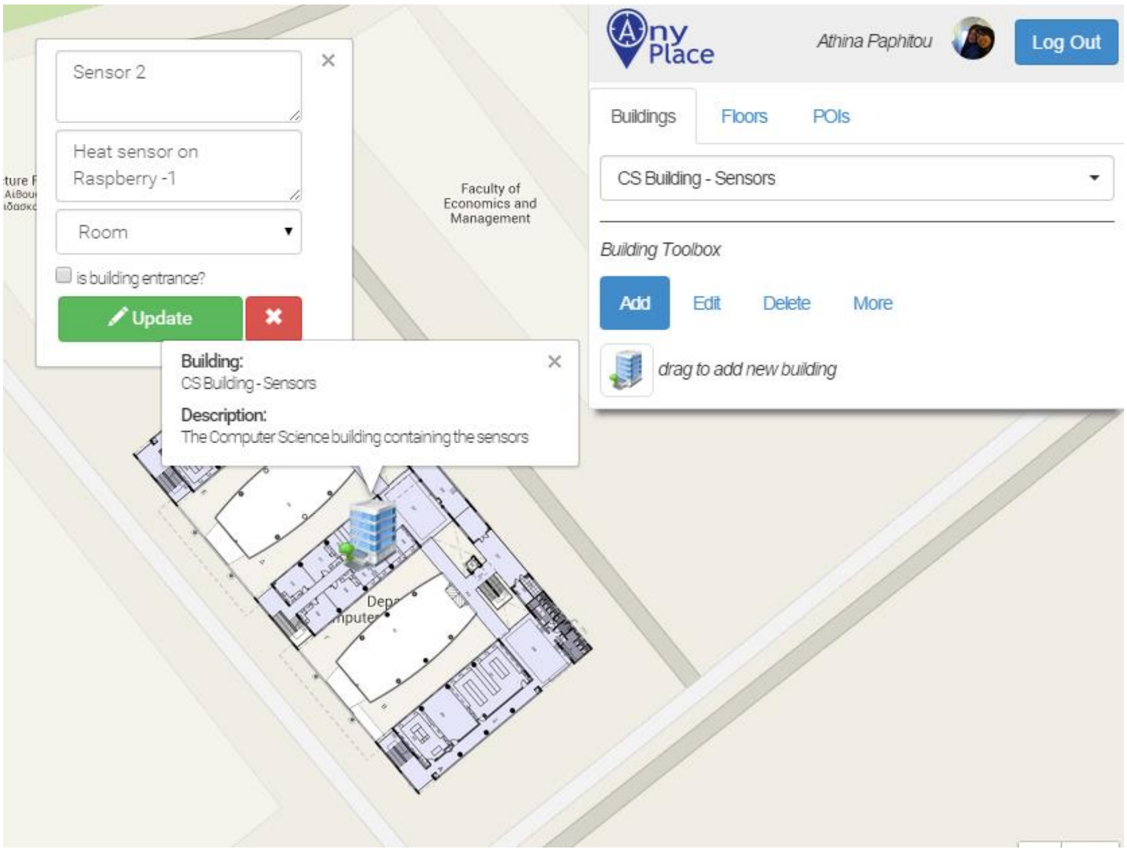
4.4 Εφαρμογή Anyplace

Στην υλοποίηση της εφαρμογής χρησιμοποιήθηκε το Anyplace για την προβολή των αισθητήρων στο χάρτη με σκοπό να βοηθήσει το χρήστη στην αναζήτηση. Το Anyplace είναι ένα crowdsourced indoor navigation σύστημα που υλοποιήθηκε από το εργαστήριο DMSL του τμήματος πληροφορικής του Πανεπιστημίου Κύπρου [5].

Συγκεκριμένα, για τους σκοπούς της ατομικής διπλωματικής εργασίας χρησιμοποιήθηκε το εργαλείο Anyplace Architect και μια κλήση το API της υπηρεσίας διαδικτύου που παρέχει το Anyplace.

4.4.1 Anyplace Architect

Το συγκεκριμένο εργαλείο επιτρέπει στο χρήστη να δημιουργήσει ένα νέο κτίριο προσθέτοντας το αρχιτεκτονικό του σχέδιο. Σε αυτό το κτίριο ο χρήστης μπορεί να προσθέσει όποια σημεία πάνω στο χάρτη θέλει ως Points Of Interest για μετέπειτα χρήση τους. Για τους σκοπούς της διπλωματικής δημιουργήθηκε το κτίριο ΘΕΕ01 και πιο συγκεκριμένα ο δεύτερος όροφος και προστέθηκαν κάποιοι αισθητήρες σαν Points of Interest στην εφαρμογή. Στη συνέχεια, με τη χρήση μιας κλήσης στην υπηρεσία διαδικτύου που προσφέρει το Anyplace, τα σημεία αυτά εμφανίζονται στο χάρτη στην οθόνη της εφαρμογής.



Εικόνα 16: Χρήση του Anyplace Architect

Πιο πάνω φαίνεται το κτίριο ΘΕΕ 01 που δημιουργήθηκε για τους σκοπούς της ατομικής διπλωματικής εργασίας και ένας αισθητήρας που τοποθετήθηκε ως σημείο ενδιαφέροντος στο συγκεκριμένο κτίριο.

4.4.2 Anyplace REST service API call

Για να γίνει η λήψη των σημείων ενδιαφέροντος στο συγκεκριμένο κτίριο χρησιμοποιήθηκε η πιο κάτω κλήση στην υπηρεσία διαδικτύου που παρέχει το Anyplace.

URL: http://anyplace.rayzit.com/anyplace/mapping/pois/all_floor

Παράμετροι: buid (building id) = building_3c4fc00b-4bd4-4c97-8314-75a4e68d07d6_1427307670430 (για το κτίριο που δημιουργήθηκε για τους σκοπούς της Α.Δ.Ε) , floor_number=2

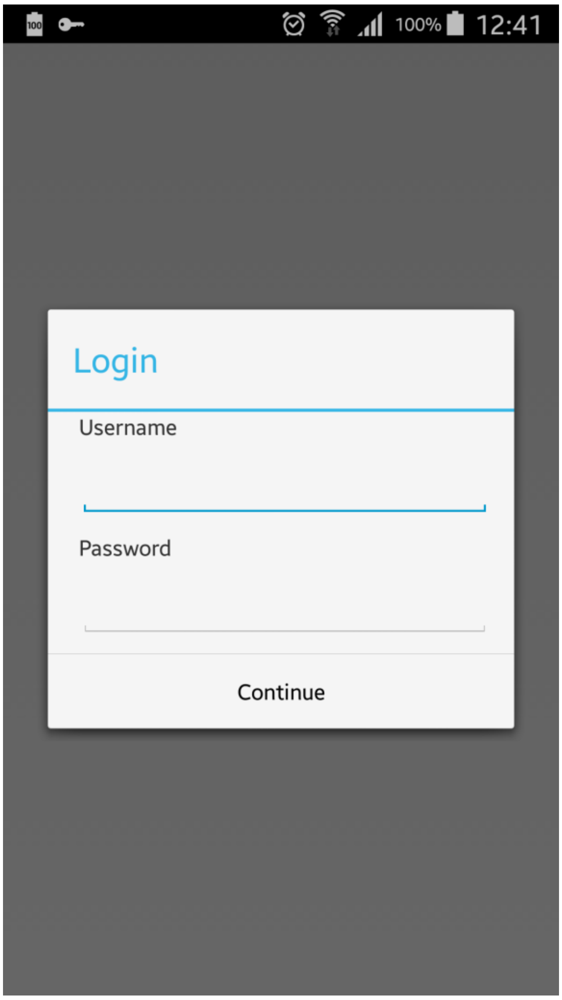
Η κλήση αυτής της υπηρεσίας διαδικτύου επιστρέφει όλα τα σημεία ενδιαφέροντος (Anyplace Points of Interest) σε μορφή JSON. Ανάμεσα στις πληροφορίες για τα σημεία ενδιαφέροντος συμπεριλαμβάνονται και οι συντεταγμένες των σημείων έτσι ώστε να γίνει η τοποθέτησή τους σε ένα χάρτη.

Περισσότερες πληροφορίες για τη χρήση του Anyplace μπορούν να βρεθούν στις παραπομπές [7][8].

5 Χρήση εφαρμογής

Σε αυτή την ενότητα παρουσιάζεται αναλυτικά η εφαρμογή με επεξήγηση των σχετικών οθονών που εμφανίζονται στο χρήστη και τις λειτουργίες που μπορούν να εκτελεστούν από κάθε οθόνη της εφαρμογής.

5.1 Είσοδος στο σύστημα



Εικόνα 17: SensoMan log-in screen

Αυτή είναι η πρώτη οθόνη που βλέπει ο χρήστης και ζητάει από αυτόν να εισάγει το όνομα και το συνθηματικό του για χρήση της εφαρμογής. Ανάλογα με αυτά, αποστέλλεται από την υπηρεσία διαδικτύου στην εφαρμογή ένα token που χρησιμοποιείται σε όλη τη διάρκεια χρήσης της εφαρμογής και καθορίζεται και ο τύπος του χρήστη που χρησιμοποιεί την εφαρμογή.

5.2 Επιλογή τοποθεσίας και παραμέτρων αναζήτησης

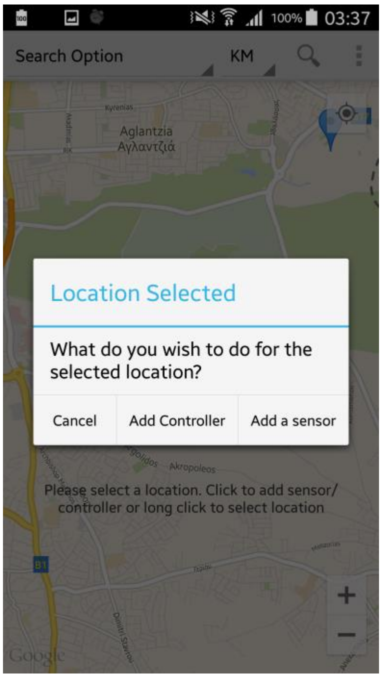
Εικόνα 18: SensoMan map screen

Η πρώτη οθόνη που βλέπει ο χρήστης μετά την είσοδό του στην εφαρμογή φαίνεται πιο πάνω. Η οθόνη αυτή χρησιμεύει στην εισαγωγή της τοποθεσίας αναζήτησης και των παραμέτρων αναζήτησης. Τα σημαντικά στοιχεία της διεπαφής επεξηγούνται πιο κάτω.

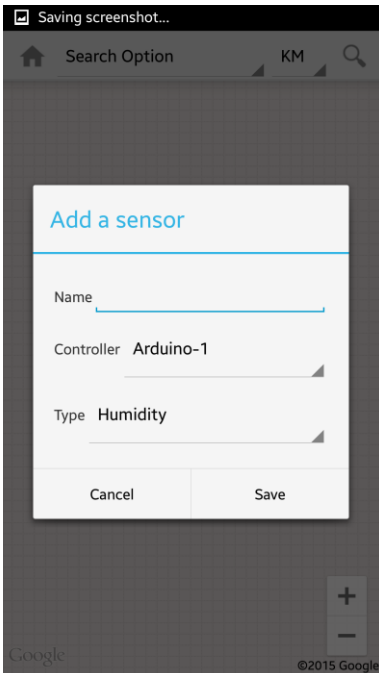
1. Εισαγωγή τύπου αναζήτησης: Μια λίστα που περιέχει όλους τους τύπους αναζήτησης που μπορεί να κάνει ο χρήστης οι οποίοι είναι : Search for sensors, search for controllers, search for sensor types.

2. Εισαγωγή ακτίνας αναζήτησης: Μια λίστα που περιέχει την ακτίνα της αναζήτησης. Η ακτίνα αυτή μπορεί να είναι 5,10,15,20 χιλιόμετρα.

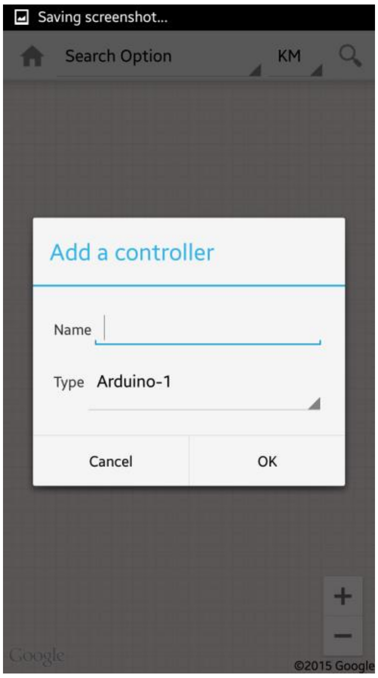
3. Επιλογή τοποθεσίας: Κάνοντας touch στο χάρτη ο χρήστης επιλέγει μια τοποθεσία έτσι ώστε να γίνει η αναζήτηση βάση αυτής ή για προσθήκη νέου αισθητήρα ή μικροελεγκτή στο σύστημα σε αυτή την τοποθεσία. Οι οθόνες για προσθήκη νέου αισθητήρα ή μικροελεγκτή φαίνονται πιο κάτω



Εικόνα 19: SensoMan location options



Εικόνα 20: SensoMan add sensor

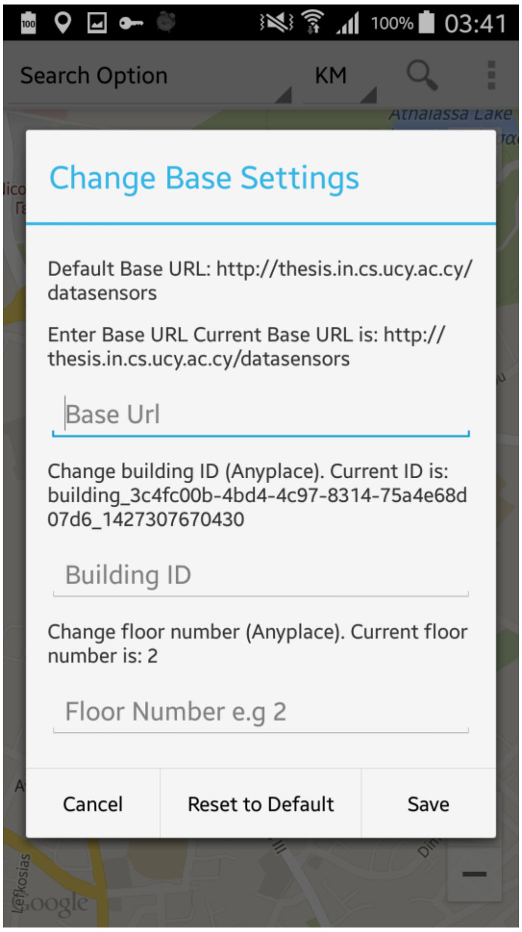


Εικόνα 21: SensoMan add controller

Όπως φαίνεται και από τις εικόνες πιο πάνω όταν ο χρήστης επιλέξει μια τοποθεσία του εμφανίζεται το σχετικό μήνυμα (Εικόνα 19). Στη συνέχεια μπορεί να προσθέσει νέο αισθητήρα εισάγοντας το όνομα του αισθητήρα, τον τύπο του και το μικροελεγκτή στον οποίο ανήκει χρησιμοποιώντας το παράθυρο που φαίνεται στην εικόνα 20 ή να εισάγει νέο μικροελεγκτή εισάγοντας το όνομα και τον τύπο του στο παράθυρο που φαίνεται στην εικόνα 21.

4. Μόλις ο χρήστης επιλέξει τοποθεσία και παραμέτρους αναζήτησης μπορεί να πατήσει το κουμπί της αναζήτησης για λήψη των αποτελεσμάτων.
5. Στην οθόνη του χάρτη φαίνεται συνεχώς η τοποθεσία του χρήστη
6. Στην οθόνη του χάρτη φαίνονται επίσης όλα τα σημεία ενδιαφέροντος που υπάρχουν στο σύστημα Anyplace.

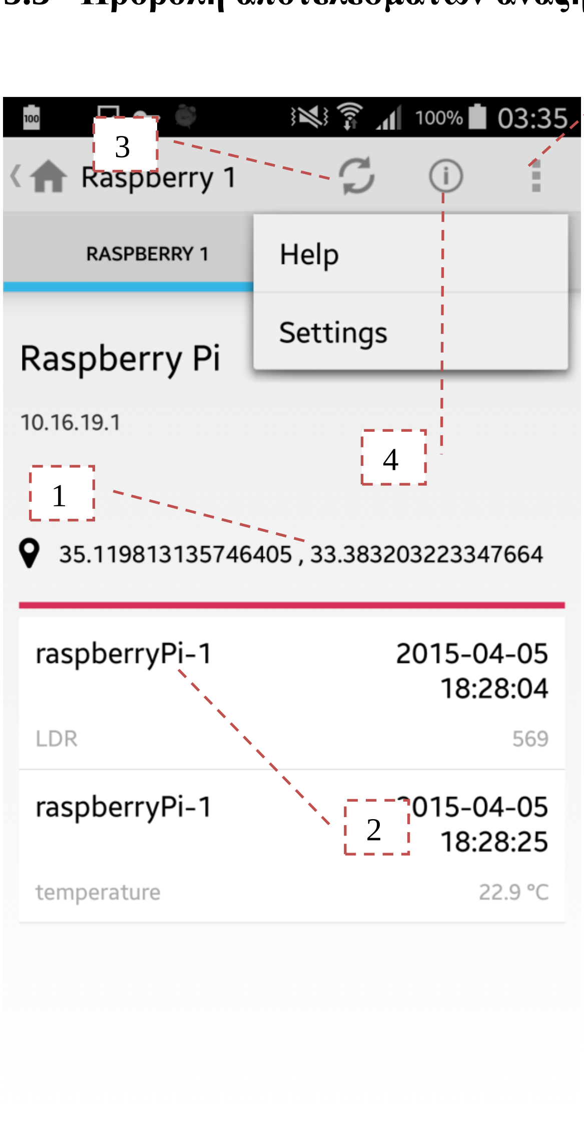
7. Με τη χρήση του menu που βρίσκεται πάνω δεξιά στην οθόνη δίνεται η δυνατότητα στο χρήστη για αποσύνδεση (Log out), να δει βοηθητικό μήνυμα (Help) ή να αλλάξει τις ρυθμίσεις που φαίνονται πιο κάτω.



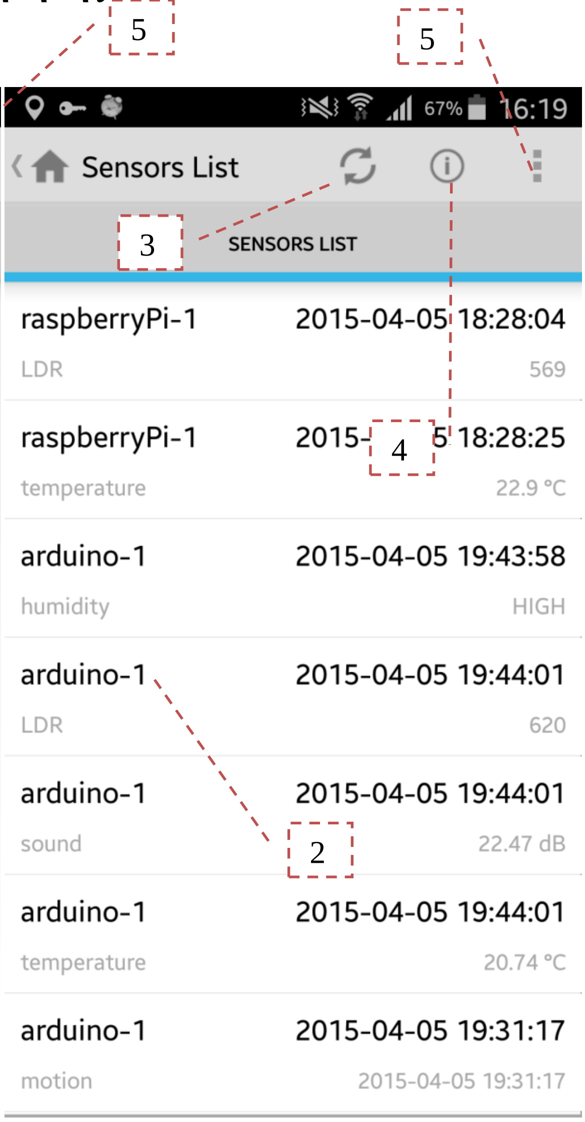
Εικόνα 22: SensoMan map screen settings

Από την οθόνη του χάρτη, ο χρήστης μπορεί να αλλάξει τα δεδομένα που φαίνονται πιο πάνω (Εικόνα 22). Αυτά περιλαμβάνουν το βασικό URL για πρόσβαση στην υπηρεσία διαδικτύου και τις παραμέτρους της υπηρεσίας Anyplace building id και floor number.

5.3 Προβολή αποτελεσμάτων αναζήτησης



Εικόνα 23: SensoMan search for controllers results

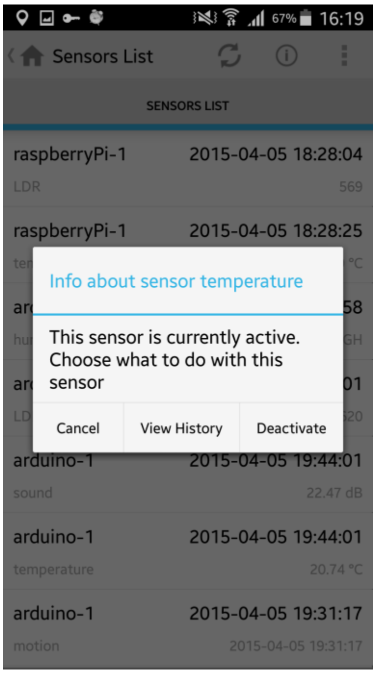


Εικόνα 24: SensoMan search for sensors/sensor types results

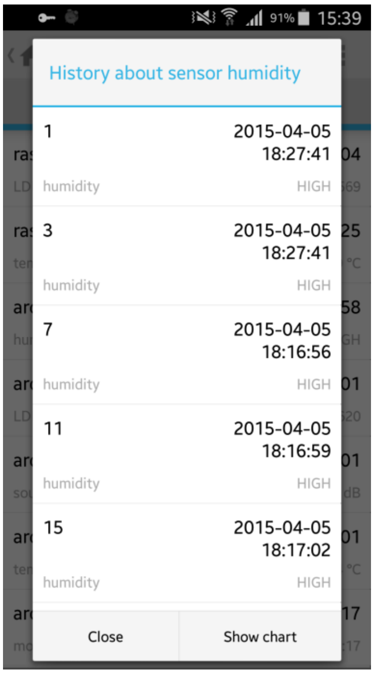
Αυτές οι δύο οθόνες χρησιμεύουν στην προβολή των αποτελεσμάτων αναζήτησης. Η πρώτη οθόνη (Εικόνα 23) είναι αυτή που βλέπει ο χρήστης όταν επιλέξει τον τύπο αναζήτησης με βάση τους μικροελεγκτές. Ο κάθε μικροελεγκτής που επιστρέφεται από την υπηρεσία διαδικτύου εμφανίζεται σε ξεχωριστό κομμάτι (με tabs) στην οθόνη. Για τον κάθε μικροελεγκτή εμφανίζονται πληροφορίες για τη τοποθεσία του, το όνομά του και τη διεύθυνση ip του (1).

Η δεύτερη οθόνη (Εικόνα 24) είναι η οθόνη που εμφανίζεται στο χρήστη στην περίπτωση αναζήτησης με βάση τους αισθητήρες ή τον τύπο των αισθητήρων. Σε αυτή την περίπτωση, τα δεδομένα δεν χωρίζονται ανά μικροελεγκτές αλλά εμφανίζονται όλα σε μια λίστα. Τα υπόλοιπα στοιχεία διεπαφής (2-5) είναι τα ίδια και για τις δύο οθόνες και επεξηγούνται αναλυτικά πιο κάτω.

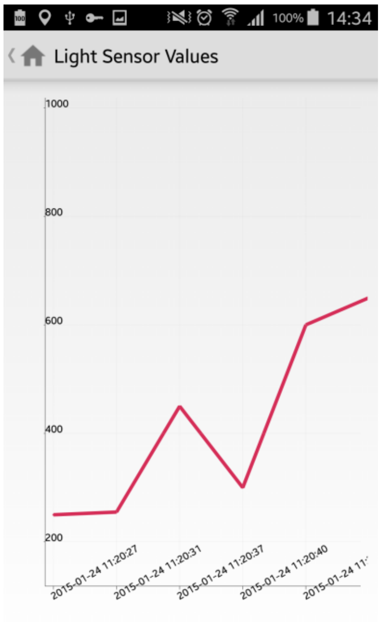
2. Η κάθε οθόνη έχει μια λίστα από αισθητήρες και τις τελευταίες μετρήσεις τους. Σε κάθε στοιχείο της λίστας βρίσκεται το όνομα του αισθητήρα, σε ποιο μικροελεγκτή είναι συνδεδεμένος ο συγκεκριμένος αισθητήρας, η ημερομηνία μέτρησης της τελευταίας τιμής και η τελευταία μέτρηση του αισθητήρα. Όταν ο χρήστης πατήσει πάνω σε ένα στοιχείο της λίστας του εμφανίζονται δύο επιλογές (Εικόνα 25) που αφορούν την απενεργοποίηση του αισθητήρα και την προβολή των ιστορικών τιμών του αισθητήρα είτε ως λίστα με παρόμοια μορφή όπως τη λίστα που φαίνεται στο σημείο 2 αλλά με τι διαφορά ότι αντί το όνομα



Εικόνα 25: SensoMan sensor options



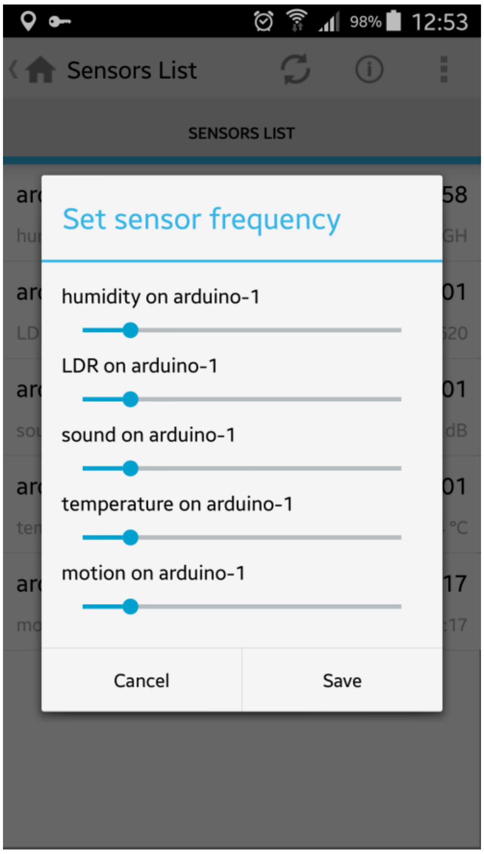
Εικόνα 26: SensoMan view history



Εικόνα 27: SensoMan view as a chart

του μικροελεγκτή φαίνεται ο αριθμός της μέτρησης (Εικόνα 26) , είτε ως γραφική παράσταση (Εικόνα 27).

3. Αυτό το κουμπί χρησιμεύει για ανανέωση των αποτελεσμάτων από την ίδια οθόνη χωρίς να χρειαστεί ο χρήστης να πάει στην προηγούμενη οθόνη και να εκτελέσει νέα αναζήτηση.
4. Αυτό το κουμπί δείχνει πληροφορίες σχετικά με τους αισθητήρες που βρίσκονται ενωμένοι εκείνη τη στιγμή των οποίων οι τιμές είναι ορατές από την οθόνη στην οποία βρίσκεται ο χρήστης.
5. Το menu στο πάνω δεξιά μέρος της οθόνης προσφέρει βοήθεια στο χρήστη και μέσω αυτού μπορεί ο χρήστης να αλλάξει τις ρυθμίσεις του μικροελεγκτή (αν βρίσκεται στην οθόνη αναζήτησης με βάση μικροελεγκτές) (Εικόνα 28) ή των αισθητήρων (αν βρίσκεται στην οθόνη αναζήτησης με βάση τους αισθητήρες) (Εικόνα 28).



Εικόνα 28: SensoMan change sensor settings

Όπως φαίνεται και από τις πιο πάνω οθόνες, ο χρήστης στην περίπτωση που κάνει αναζήτηση με βάση τους μικροελεγκτές μπορεί να αλλάξει τις ρυθμίσεις του κάθε μικροελεγκτή απενεργοποιώντας τον ή αλλάζοντας τη συχνότητα με την οποία μαζεύει δεδομένα (Εικόνα 28). Από την άλλη μεριά, αν ο χρήστης αναζητήσει με βάση αισθητήρες ή τύπους αισθητήρων, έχει τη δυνατότητα να αλλάξει τη συχνότητα του κάθε αισθητήρα ξεχωριστά (Εικόνα 29).

6 Αρχική Αξιολόγηση από Χρήστες

Σε αυτό το κεφάλαιο γίνεται αναφορά στον τρόπο αξιολόγησης της κινητής εφαρμογής SensoMan που υλοποιήθηκε στα πλαίσια της ΑΔΕ. Για σκοπούς ελέγχου της εφαρμογής για σφάλματα διενεργήθηκε μιας μικρής κλίμακας beta testing ενώ για σκοπούς αξιολόγησης της ευχρηστίας και χρησιμότητας της εφαρμογής διενεργήθηκε διαδικασία αξιολόγησης με ερωτηματολόγιο. Ο έλεγχος της εφαρμογής για σφάλματα πραγματοποιήθηκε και με τη βοήθεια άλλων προπτυχιακών φοιτητών του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου. Το ερωτηματολόγιο στάλθηκε σε προπτυχιακούς και μεταπτυχιακούς φοιτητές του Πανεπιστημίου Κύπρου σε μορφή Google forms για εύκολη εξαγωγή αποτελεσμάτων και στατιστικών στοιχείων με γραφικές παραστάσεις.

Αξίζει να σημειωθεί ότι η συγκεκριμένη αξιολόγηση από χρήστες της εφαρμογής έγινε σε περιορισμένο βαθμό. Συγκεκριμένα, έγινε μόνο μερική αξιολόγηση σε θέματα αποδοχής της εφαρμογής από χρήστες αλλά και σε γενικά θέματα ευχρηστίας. Η εφαρμογή δεν έλαβε εκτενούς αξιολόγησης λόγω του ότι υπήρχαν κάποιοι περιορισμοί στη χρήση της εφαρμογής όπως, για παράδειγμα, το γεγονός ότι για να γίνει χρήση της εφαρμογής απαιτείται η σύνδεση με το δίκτυο του τμήματος Πληροφορικής για να γίνει λήψη δεδομένων από την υπηρεσία διαδικτύου. Αυτό οδήγησε στη διεξαγωγή μιας μικρής κλίμακας αξιολόγησης τόσο σε περιεχόμενο όσο και σε αριθμό χρηστών που έλαβαν μέρος.

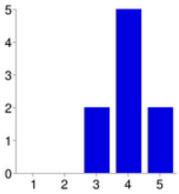
6.1 Ερωτηματολόγιο Αξιολόγησης

Για σκοπούς αξιολόγησης της αποδοχής της εφαρμογής από τους χρήστες, δημιουργήθηκε ένα ερωτηματολόγιο με Google forms που περιλαμβάνει ερωτήσεις σχετικά με την ευχρηστία της εφαρμογής και ζητάει από το χρήστη να αξιολογήσει το κατά πόσο βρήκε την εφαρμογή χρήσιμη. Οι ερωτήσεις του ερωτηματολογίου φαίνονται αναλυτικά στο παράρτημα Γ. Το ερωτηματολόγιο μπορεί επίσης να βρεθεί μέσω ενός συνδέσμου που βρίσκεται στη βιβλιογραφία [15].

6.2 Αποτελέσματα αξιολόγησης

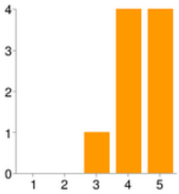
Μέσω του Google Forms τα αποτελέσματα που λήφθηκαν από 9 άτομα που συμπλήρωσαν το ερωτηματολόγιο εμφανίζονται σε συνοπτική μορφή με τη βοήθεια γραφικών παραστάσεων. Λόγω περιορισμών όπως το γεγονός ότι για να λειτουργήσει η εφαρμογή απαιτείτο η σύνδεση στο δίκτυο του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου, η αξιολόγηση από τους χρήστες έγινε σε περιορισμένο βαθμό. Τα αποτελέσματα της αξιολόγησης παρατίθενται αυτούσια πιο κάτω.

Try to search for controllers. How easy was it for you to get any results?



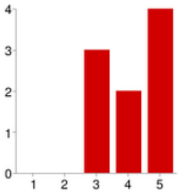
1	**0**	0%
2	**0**	0%
3	**2**	22.2%
4	**5**	55.6%
5	**2**	22.2%

Try to search for sensors. How easy was it for you to get any results?



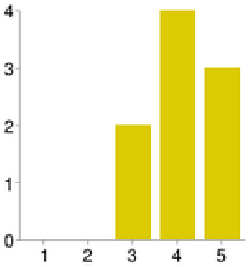
1	**0**	0%
2	**0**	0%
3	**1**	11.1%
4	**4**	44.4%
5	**4**	44.4%

Try to change some settings in the map screen. How easy was it for you to understand what settings can you change and actually change them?



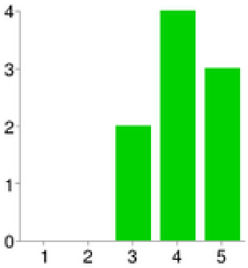
1	**0**	0%
2	**0**	0%
3	**3**	33.3%
4	**2**	22.2%
5	**4**	44.4%

As an admin or sensor owner try to change a sensor's frequency. How easy was it for you to do that?



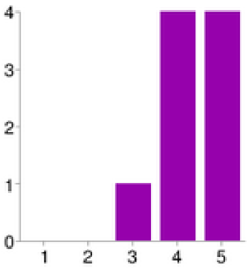
1	**0**	0%
2	**0**	0%
3	**2**	22.2%
4	**4**	44.4%
5	**3**	33.3%

As an admin or sensor owner try to change a controller's frequency. How easy was it for you to do that?



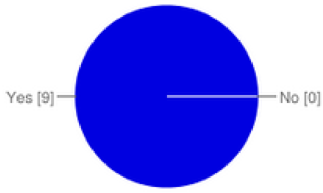
1	**0**	0%
2	**0**	0%
3	**2**	22.2%
4	**4**	44.4%
5	**3**	33.3%

Try to view a sensor's history values. How easy was it for you to to that?



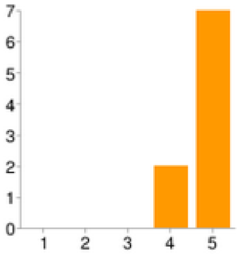
1	**0**	0%
2	**0**	0%
3	**1**	11.1%
4	**4**	44.4%
5	**4**	44.4%

Try to view a sensor's history values displayed as a chart. Were you able to read and understand the chart?



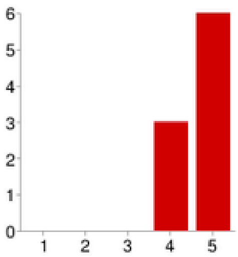
Yes	**9**	100%
No	**0**	0%

Try to deactivate a sensor. How easy was it for you to do that?



1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**2**	22.2%
5	**7**	77.8%

Try to add a new controller or sensor. How easy was it for you to do that?



1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**3**	33.3%
5	**6**	66.7%

Did you find the completion of any of the tasks above particularly difficult? If so please state the task and the reasons that made it difficult for you to complete it.

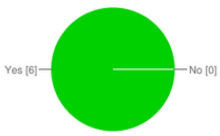
Just the first time the change of settings. After that it's all okay

Part 2 - General questions

Did you understand the concept of the application?

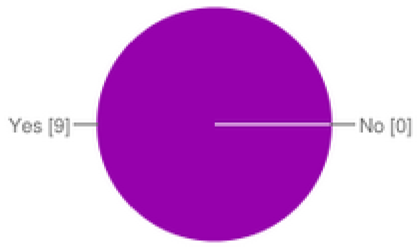
Yes	**6**	100%
No	**0**	0%

Do you think that this application is a good way to manage your sensors if you ever want to?



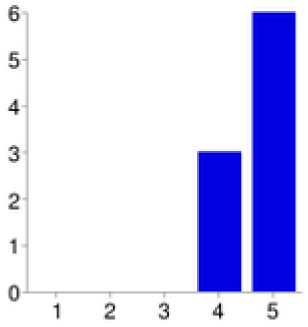
Yes	**6**	100%
No	**0**	0%

Did you know where to find help if you needed to?



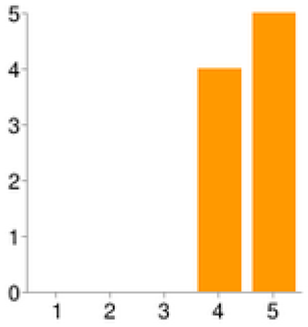
Yes	9	100%
No	0	0%

How easy was it for you to navigate between different screens?



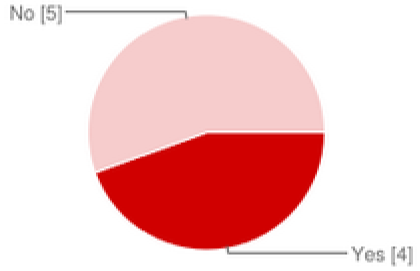
1	0	0%
2	0	0%
3	0	0%
4	3	33.3%
5	6	66.7%

How helpful were the on-screen instructions and/or help messages?



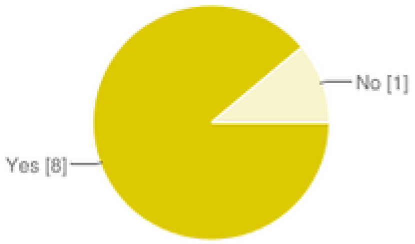
1	0	0%
2	0	0%
3	0	0%
4	4	44.4%
5	5	55.6%

Did you use the "About" button at all?



Yes	4	44.4%
No	5	55.6%

On the map screen could you find any Anyplace POIs?



Yes	8	88.9%
No	1	11.1%

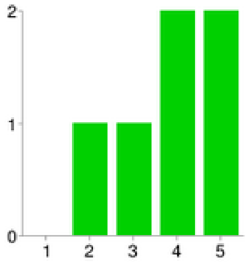
Leave any suggestions/concerns about the functionality of the application

in some cases no location was selected and it still produced results

Did the application crash at any point? If so please state the steps you took and the screen you were on.

no
Nope

In your opinion, how useful could this application be to you?



1	**0**	0%
2	**1**	16.7%
3	**1**	16.7%
4	**2**	33.3%
5	**2**	33.3%

Can you think of a situation that this application could be applied to?

Tha mporouse na xrisimopoiithoun gia tin asfaleia enos mouseiou xrisimopiontas tous sensors.
Smart home
Manage the observation of different sensors in nature to identify conflagration, earthquakes and other natural disasters
manage the security of your properties
Manage the sensors of an automated security system
manage sensors of my home

Εικόνα 29: Απαντήσεις ερωτηματολογίου

6.3 Συμπεράσματα αποτελεσμάτων

Παρατηρώντας τα αρχικά αποτελέσματα από τις απαντήσεις του ερωτηματολογίου βρέθηκαν κάποια μικρά σφάλματα στη λειτουργία της εφαρμογής όπως για παράδειγμα το ότι η εφαρμογή σε ορισμένες περιπτώσεις έβγαζε μετρήσεις από αισθητήρες χωρίς να επιλέξει ο χρήστης τοποθεσία ή παραμέτρους αναζήτησης. Αυτά τα σφάλματα διορθώθηκαν σε μετέπειτα στάδιο κατά τη διάρκεια της διεξαγωγής της αξιολόγησης.

Ακόμα ένα ενδιαφέρον συμπέρασμα που εξήχθηκε από την διεξαγωγή της αξιολόγησης ήταν ότι οι χρήστες που χρησιμοποιούσαν την εφαρμογή για πρώτη φορά έβρισκαν δυσκολία στη διεξαγωγή αναζήτησης. Αυτό οδήγησε στη δημιουργία ενός οδηγού χρήσης στην εφαρμογή που εμφανίζεται στο χρήστη όταν τρέξει για πρώτη φορά την εφαρμογή μετά την εγκατάστασή της. Επιπλέον, τα αποτελέσματα της έρευνας οδήγησαν και στον εμπλουτισμό των βοηθητικών μηνυμάτων στην εφαρμογή. Παρόλα αυτά όμως, ακόμα και μετά την εισαγωγή των επιπλέον βοηθητικών μηνυμάτων και του οδηγού χρήσης της εφαρμογής, μερικοί χρήστες εξακολουθούσαν να έχουν δυσκολία στα αρχικά στάδια χρήσης της εφαρμογής αφού πολύ λίγοι αφιέρωναν χρόνο στο να διαβάσουν τον οδηγό χρήσης ή να ψάξουν για βοήθεια.

Όσο αφορά τη χρήση του Anyplace σχεδόν όλοι οι χρήστες κατάφεραν να εντοπίσουν τα σημεία ενδιαφέροντος (Anyplace Points of Interest) πράγμα που σημαίνει ότι η χρήση της υπηρεσίας Anyplace θα ήταν ένας καλός τρόπος προσθήκης όλων των αισθητήρων στο χάρτη.

Τέλος, σε σχέση με τον έλεγχο της χρησιμότητας της εφαρμογής, όλοι οι χρήστες ήταν σε θέση να καταλάβουν το σκοπό χρήσης της εφαρμογής και όλοι δήλωσαν πώς η εφαρμογή SensoMan αποτελεί ένα καλό τρόπο διαχείρισης αισθητήρων σε περίπτωση που αυτό είναι απαραίτητο. Σχετικά με την ερώτηση «Πόσο χρήσιμη μπορεί να είναι αυτή η εφαρμογή για εσάς» το 66.6% απάντησε χρήσιμη ως πολύ χρήσιμη ενώ το 33.3% απάντησε λίγο ως μέτρια χρήσιμη.

Στην ερώτηση που αφορούσε πιθανές χρήσεις της εφαρμογής λήφθηκαν ενδιαφέρουσες απαντήσεις όπως:

- Smart home
- Διαχείριση αισθητήρων σε ένα σύστημα ασφαλείας
- Διαχείριση αισθητήρων που βρίσκονται έξω στη φύση
- Διαχείριση αισθητήρων στο σπίτι

Οι απαντήσεις αυτές συμβαδίζουν με το σκοπό ανάπτυξης της εφαρμογής και δείχνουν ότι η εφαρμογή θα μπορούσε να χρησιμοποιηθεί σε πολλούς τομείς στους οποίους η διαχείριση των αισθητήρων αποτελεί αναγκαία.

7 Κεφάλαιο Συμπεράσματα και Μελλοντική ανάπτυξη

7.1 Συμπεράσματα

Μετά από τη διεξαγωγή της αξιολόγησης και τη λήψη των αποτελεσμάτων, φάνηκε ότι η εφαρμογή SensoMan είναι ένας καλός τρόπος επίτευξης του σκοπού της ΑΔΕ. Η δύναμη της εφαρμογής οφείλεται στο γεγονός ότι μπορεί να χρησιμοποιηθεί με οποιοδήποτε είδος ελεγκτή, φτάνει ο ελεγκτής (δηλαδή η συσκευή στην οποία είναι συνδεδεμένοι οι αισθητήρες) να έχει την δυνατότητα πρόσβασης στο διαδίκτυο και της αποστολής HTTP requests. Μέσω αυτού του τρόπου επικοινωνίας, εξασφαλίζεται μια αφαιρετικότητα και ευελιξία υπό την έννοια ότι η εφαρμογή μπορεί να έχει εύκολα πρόσβαση σε οποιαδήποτε υπηρεσία διαδικτύου και συνεπώς επιτρέπει στο χρήστη να δημιουργήσει τη δική του υπηρεσία διαδικτύου για τα δικά του δεδομένα.

7.2 Μελλοντική ανάπτυξη

Ένας από τους στόχους για μελλοντική ανάπτυξη της εφαρμογής είναι η διεξαγωγή μιας πιο εκτενούς αξιολόγησης από τους τελικούς χρήστες. Συγκεκριμένα, η εφαρμογή θα πρέπει να εξεταστεί και σε θέματα απόδοσης (όπως π.χ. χρόνοι απόκρισης) αλλά και σε παιραιτέρω θέματα που έχουν να κάνουν με την αλληλεπίδραση των χρηστών με την εφαρμογή.

Επίσης, στα πλαίσια της μελέτης περί μελλοντικής ανάπτυξης της εφαρμογής, εξετάστηκε η περίπτωση χρήσης του μικροελεγκτή Raspberry Pi ή μιας εφαρμογής Android ως κάποιου είδους τηλεχειριστηρίο για διάφορες συσκευές. Δηλαδή, εξετάστηκε αν υπάρχει η δυνατότητα χρήσης του Raspberry Pi για ενεργοποίηση ή απενεργοποίηση οικιακών και άλλων συσκευών μέσω κάποιου μηχανισμού, βάση των μετρήσεων που παίρνει από αισθητήρες. Τα αποτελέσματα αυτής της έρευνας ανέδειξαν δύο εταιρίες που προσπαθούν να επιτεύξουν αυτό το σκοπό μέσω κάποιων συσκευών που παρουσιάζονται πιο κάτω.

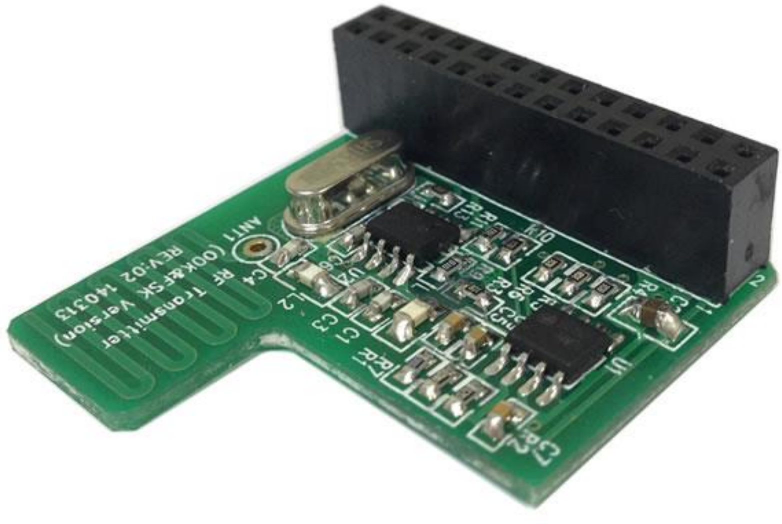
7.2.1 Linkio

Η εταιρία Linkio δημιούργησε ένα hub το οποίο συνδέεται με το διαδίκτυο και μπορεί να διαβάζει σήματα από συμβατές συσκευές [16]. Οι συσκευές αυτές περιλαμβάνουν ένα ειδικό είδος πρίζας που ενώνεται με την παροχή ρεύματος, ένα διακόπτη φωτός και διάφορους αισθητήρες που έχουν τη δυνατότητα να στέλνουν και να λαμβάνουν σήματα μέσω wi-fi. Επίσης, δημιούργησαν ένα remote που διαβάζει σήματα από μια εφαρμογή Android για ενεργοποίηση και απενεργοποίηση των συσκευών και αισθητήρων που είναι συνδεδεμένοι με τις ειδικές συσκευές.
Τέλος, δίνουν τη δυνατότητα σε developers να συνδέσουν όποιου είδους μικροελεγκτή μέσω του Developer’s Kit που έχουν υλοποιήσει [17]. Το developer’s kit θα παρέχει APIs και βιβλιοθήκες που θα βοηθούν στη λήψη δεδομένων αλλά και στην επικοινωνία με το ειδικό hub. Επίσης, θα υποστηρίζει γλώσσες όπως C/C++ κάνοντας έτσι δυνατή την χρήση ενός Raspberry Pi ή Arduino ως master και του hub ως slave.

Αξίζει να σημειωθεί ότι η συγκεκριμένη εταιρία είναι ακόμα στα αρχικά στάδια και δεν υπάρχει κάτι χειροπιαστό μέχρι τώρα.

7.2.2 Energenie Pi-mote

Η εταιρία αυτή υλοποίησε ένα shield με το όνομα Pi-mote ειδικά για το Raspberry Pi που έχει τη δυνατότητα να επικοινωνεί με τις ειδικές συσκευές που κατασκευάζουν για έλεγχο διάφορων οικιακών συσκευών [18][19]. Το συγκεκριμένο shield δίνει τη δυνατότητα ελέγχου συσκευών από το Raspberry Pi οι οποίες είναι ενωμένες με τα ειδικά sockets που κατασκευάζει η Energenie. Για το συγκεκριμένο shield υπάρχει αρκετό documentation στο διαδίκτυο [20] καθώς και παραδείγματα χρήσης του στη σελίδα του Raspberry Pi [21] αλλά και από πολλούς χρήστες. Συνεπώς, η συγκεκριμένη υλοποίηση έχει ελεγχθεί σε μεγάλο βαθμό και υπάρχει αρκετή υποστήριξη στο διαδίκτυο. Το συγκεκριμένο shield φαίνεται πιο κάτω (Εικόνα 36).



Εικόνα 30: Pi-mote από Energenie

Όπως περιγράφεται στην τεκμηρίωση του shield, δίνεται η δυνατότητα στο χρήστη μέσω Python να χρησιμοποιήσει τις δυνατότητες που του προσφέρει το Pi-mote. Οπόταν, για μετατροπή του Raspberry Pi σε αυτόματο τηλεχειριστήριο με βάση τιμές από αισθητήρες, το μόνο που χρειάζεται είναι η κλήση των μεθόδων που παρέχει το Pi-mote ανάλογα με τις τιμές που διαβάζει το Raspberry Pi από τους αισθητήρες μέσω Python. Το μόνο πρόβλημα που υπάρχει είναι ότι το συγκεκριμένο shield φαίνεται να καλύπτει όλες τις διαθέσιμες GPIO pins του μικροελεγκτή και συνεπώς θα πρέπει ο μικροελεγκτής να λαμβάνει τις μετρήσεις από τους αισθητήρες με άλλο τρόπο (σύνδεση με ένα Arduino ή μέσω μιας υπηρεσίας διαδικτύου).

Βιβλιογραφία

[1] http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/LDR-Datasheet.pdf

[2] http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/159-282-86001.pdf

[3] http://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge

[4] Android https://developer.android.com/guide/index.html

[5] "Συλλογή δεδομένων αισθητήρων μέσω της πλατφόρμας Arduino και χειρισμός τους μέσω μηχανικής μάθησης", Στέλλα Κωνσταντίνου, Μάιος 2015

[6] Anyplace: A Crowdsourced Indoor information System", K. Georgiou, T. Constambeys, C. Laoudias, L. Petrou, G. Chatzimilioudis and D. Zeinalipour-Yazti (MDM'15), Pittsburgh, Pennsylvania, USA, June 15-18, 2015

[7] http://anyplace.cs.ucy.ac.cy/developers/

[8] Laoudias, C., Constantinou, G., Constantinides, M., Nicolaou, S., Zeinalipour-Yazti, D., and Panayiotou, C. G. 2012. The airplace indoor positioning platform for android smartphones. In Proceedings of the IEEE 13th International Conference on Mobile Data Management, MDM ’12, 312-315). IEEE.

[9] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. 1999. Towards a better understanding of context and context-awareness. Handheld and ubiquitous computing, 304-307. Springer Berlin Heidelberg.

[10] https://play.google.com/store/apps/details?id=fabrizio.boco.arduinoManager&hl=en

[11] https://play.google.com/store/apps/details?id=org.connectbot&hl=en

[12] https://play.google.com/store/apps/details?id=com.realvnc.viewer.android&hl=en

[13] https://code.google.com/p/achartengine/

[14] https://github.com/amlcurran/ShowcaseView

[15] https://github.com/amlcurran/ShowcaseView

[16] http://linkio.net/en/connected-objects-solutions/

[17] http://linkio.net/en/developpers/

[18] https://www.raspberrypi.org/controlling-electrical-sockets-with-energenie-pi-mote/

[19] https://energenie4u.co.uk/index.php/catalogue/product/ENER002-2PI

[20] http://pythonhosted.org/energenie/

[21] https://www.raspberrypi.org/

[22] http://whatis.techtarget.com/definition/Internet-of-Things

8 Παράρτημα Α – Κώδικας εφαρμογής SensoMan

8.1 LocationChooserActivity class

```
package com.example.tabbedmonitor;

import java.io.BufferedInputStream;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.LinkedList;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Semaphore;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.github.amlcurran.showcaseview.*;
import com.github.amlcurran.showcaseview.targets.ActionItemTarget;
import com.github.amlcurran.showcaseview.targets.ActionViewTarget;
import
com.github.amlcurran.showcaseview.targets.ActionViewTarget.Type;
import com.github.amlcurran.showcaseview.targets.ViewTarget;
import com.google.android.gms.common.ConnectionResult;
```

```java
import
com.google.android.gms.common.GooglePlayServicesNotAvailableException;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.MapsInitializer;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.content.SharedPreferences;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;

import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemSelectedListener;

public class LocationChooserActivity extends Activity implements
        LocationListener {
```

```java
	static final LatLng NewCampus = new LatLng(35.145, 33.409);
	MapView mapView;
	protected static GoogleMap map;
	final String[] options = { "Search Option", "Search for 
sensors",
			"Search for controllers", "Search for sensor types" 
};
	final String[] radius_opt = { "KM", "5", "10", "15", "20" };
	String[] type_opt = { "Sensor Type", "Humidity", "LDR", 
"Motion", "Sound",
			"Temperature" };
	private Spinner SpinSensorType;
	static String TOKEN = null;
	public String option = null;
	public static int radius = -1;
	public String sensorType = null;
	static String resultSearch = null;
	static LatLng selected = null;
	ProgressDialog progDailog;
	static boolean firstTimeLocation;

	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		selected = null;
		firstTimeLocation = true;
		TOKEN = Utilities.applicationUser.getToken();
		Log.i("Response", Utilities.applicationUser.toString());
		setContentView(R.layout.activity_location_chooser);

		getActionBar().setTitle("");
		getActionBar().setDisplayShowHomeEnabled(false);

		try {
			type_opt = new GetSensorTypes().execute("").get();
		} catch (InterruptedException e1) {

			e1.printStackTrace();
		} catch (ExecutionException e1) {
```

```
				e1.printStackTrace();
			}
			if (type_opt == null) {
				type_opt = new String[] { "Select a sensor type",
"Humidity",
						"LDR", "sound", "temperature", "motion"
};
			}

			SpinSensorType = (Spinner)
findViewById(R.id.spinnerLocChooserType);
			ArrayAdapter<String> a2 = new ArrayAdapter<String>(
					LocationChooserActivity.this,
					android.R.layout.simple_spinner_dropdown_item,
type_opt);
			SpinSensorType.setAdapter(a2);
			SpinSensorType.setOnItemSelectedListener(new
OnItemSelectedListener() {

				@Override
				public void onItemSelected(AdapterView<?> arg0, View
arg1, int pos,
						long arg3) {
					if (pos == 0) {
						sensorType = null;
					} else {
						sensorType = type_opt[pos];
					}

				}

				@Override
				public void onNothingSelected(AdapterView<?> arg0) {

				}

			});

			try {
```

```
			MapsInitializer.initialize(getApplicationContext());
		} catch (GooglePlayServicesNotAvailableException e) {
			Log.e("Address Map", "Could not initialize google 
play", e);
		}

		switch (GooglePlayServicesUtil

	.isGooglePlayServicesAvailable(getApplicationContext())) {
		case ConnectionResult.SUCCESS:

			mapView = (MapView) findViewById(R.id.map);
			mapView.onCreate(savedInstanceState);
			// Gets to GoogleMap from the MapView and does 
initialization stuff
			if (mapView != null) {
				map = mapView.getMap();

			}
			break;
		case ConnectionResult.SERVICE_MISSING:
			Toast.makeText(getApplicationContext(), "SERVICE 
MISSING",
					Toast.LENGTH_SHORT).show();
			break;
		case ConnectionResult.SERVICE_VERSION_UPDATE_REQUIRED:
			Toast.makeText(getApplicationContext(), "UPDATE 
REQUIRED",
					Toast.LENGTH_SHORT).show();
			break;
		default:
			Toast.makeText(
					getApplicationContext(),
					GooglePlayServicesUtil

	.isGooglePlayServicesAvailable(getApplicationContext()),
					Toast.LENGTH_SHORT).show();
		}

		map.setMyLocationEnabled(true);
```

```
		LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
		Criteria criteria = new Criteria();
		String bestProvider =
locationManager.getBestProvider(criteria, true);
		Location location =
locationManager.getLastKnownLocation(bestProvider);
		if (location != null) {
			onLocationChanged(location);
		}
		locationManager.requestLocationUpdates(bestProvider,
20000, 0, this);
		map.setOnMapLongClickListener(new
GoogleMap.OnMapLongClickListener() {

			@Override
			public void onMapLongClick(LatLng arg0) {
				map.addMarker(new
MarkerOptions().position(arg0).title(
						"Selected Location"));

				selected = arg0;
				Toast.makeText(getApplicationContext(),
						selected.latitude + " " +
selected.longitude,
						Toast.LENGTH_SHORT).show();
			}
		});
		map.setOnMapClickListener(new
GoogleMap.OnMapClickListener() {

			@Override
			public void onMapClick(LatLng arg0) {

				AlertDialog.Builder builder = new
AlertDialog.Builder(
						LocationChooserActivity.this);
				builder.setTitle("Location Selected");
				builder.setMessage("What do you wish to do for
the selected location?");
```

```
                    builder.setPositiveButton("Add a sensor",
                            new OnClickListener() {

                                @Override
                                public void
onClick(DialogInterface dialog,
                                        int which) {
                                    if
(Utilities.applicationUser.getUserType() == UserTypes.simple) {
                                        Toast.makeText(

      getApplicationContext(),
                                                "You
do not have permission to change settings",

      Toast.LENGTH_LONG).show();
                                        dialog.cancel();
                                        return;
                                    }

                                    String[] types =
type_opt;
                                    LayoutInflater
inflater = getLayoutInflater();
                                    View dialoglayout =
inflater.inflate(

      R.layout.add_new_sensor, null);
                                    final Spinner
typesSpinner = (Spinner) dialoglayout

      .findViewById(R.id.spinnerAddType);

                                    ArrayAdapter<String>
adapterTypes = new ArrayAdapter<String>(

      LocationChooserActivity.this,

      android.R.layout.simple_spinner_dropdown_item,
                                            types);
```

```
		typesSpinner.setAdapter(adapterTypes);
								final EditText
sensorName = (EditText) dialoglayout

		.findViewById(R.id.editTextAddName);

								AlertDialog.Builder
builder = new AlertDialog.Builder(

		LocationChooserActivity.this);

		builder.setView(dialoglayout);

		builder.setCancelable(true);
								builder.setTitle("Add
a sensor");

		builder.setPositiveButton("Save",
											new
OnClickListener() {


		@Override

		public void onClick(

			DialogInterface dialog,

			int which) {

		UploadSensor us = new UploadSensor();

		String name = sensorName

				.getText().toString();

		int id = typesSpinner

				.getSelectedItemPosition();
```

```
        us.execute(id + "", name);

        Toast.makeText(

                    getApplicationContext(),

                    "Sensor Uploaded Successfully",

                    Toast.LENGTH_LONG)

                    .show();

        return;
                                                                       }
                                                                 });

        builder.setNegativeButton("Cancel",
                                                                 new
OnClickListener() {


        @Override

        public void onClick(

              DialogInterface dialog,

              int which) {

        return;

                                                                       }
                                                                 });
                                                      builder.show();
                                                 }
                                            });
                               builder.setNegativeButton("Cancel", new
OnClickListener() {
```

```
                                @Override
                                public void onClick(DialogInterface
dialog, int which) {

                                      dialog.dismiss();
                                }
                          });
                          builder.setNeutralButton("Add Controller",
                                      new OnClickListener() {

                                            @Override
                                            public void
onClick(DialogInterface dialog,
                                                        int which) {
                                                  if
(Utilities.applicationUser.getUserType() == UserTypes.simple) {
                                                        Toast.makeText(

      getApplicationContext(),
                                                                    "You
do not have permission to change settings",

      Toast.LENGTH_LONG).show();
                                                        dialog.cancel();
                                                        return;
                                                  }

                                                  String[] controllers =
{ "Arduino-1",
                                                              "Raspberry
Pi-1" };
                                                  LayoutInflater
inflater = getLayoutInflater();
                                                  View dialoglayout =
inflater.inflate(

      R.layout.add_controller, null);
                                                  Spinner
controllersSpinner = (Spinner) dialoglayout
```

```
        .findViewById(R.id.spinnerAddControllerType);
                                              ArrayAdapter<String>
adapterTypes = new ArrayAdapter<String>(

        LocationChooserActivity.this,

        android.R.layout.simple_spinner_dropdown_item,

        controllers);

        controllersSpinner.setAdapter(adapterTypes);
                                              AlertDialog.Builder
builder = new AlertDialog.Builder(

        LocationChooserActivity.this);

        builder.setView(dialoglayout);

        builder.setCancelable(true);
                                              builder.setTitle("Add
a controller");

        builder.setPositiveButton("OK",
                                                          new
OnClickListener() {


        @Override

        public void onClick(

              DialogInterface dialog,

              int which) {

        dialog.dismiss();
                                                                }
                                                          });
```

```
        builder.setNegativeButton("Cancel",
                                                                        new
OnClickListener() {

        @Override
        public void onClick(
                DialogInterface dialog,
                int which) {
        dialog.dismiss();
                                                                            }
                                                                    });
                                                        builder.show();
                                                }
                                        });
                                builder.show();
                        }
                });
                GetPOIsAnyplace gpa = new GetPOIsAnyplace();
                gpa.execute("");

        }

        private void showTutorial() {

                new ShowcaseView.Builder(this,
false).setContentTitle("Welcome")
                                .setContentText("Long click on the map to
select a location.")

        .setStyle(R.style.CustomShowcaseView).hideOnTouchOutside()
                                .setShowcaseEventListener(new
OnShowcaseEventListener() {
```

```
                              @Override
                              public void
onShowcaseViewShow(ShowcaseView showcaseView) {

                              }

                              @Override
                              public void
onShowcaseViewHide(ShowcaseView showcaseView) {

                                    new
ShowcaseView.Builder(LocationChooserActivity.this)
                                                .setTarget(
                                                            new
ActionItemTarget(

      LocationChooserActivity.this,

      R.id.spinnerType))
                                                .setContentTitle("How
to get results")
                                                .setContentText(
                                                            "Select a
search option from the list e.g \"Search for controllers\".")
                                                // "\nThen, select a
search radius from the next list e.g 5km.\nFinally,click the search
button to get the results.\nNote: Click on the menu on the top right
of the screen to get more help, change settings or log-out from the
application. ")
                                                .hideOnTouchOutside()

      .setShowcaseEventListener(
                                                            new
OnShowcaseEventListener() {


      @Override

      public void onShowcaseViewShow(
```

```java
        ShowcaseView showcaseView) {
// TODO Auto-generated method
// stub
}

@Override
public void onShowcaseViewHide(
        ShowcaseView showcaseView) {
showTutorial2ndStep();
}

@Override
public void onShowcaseViewDidHide(
        ShowcaseView showcaseView) {
// TODO Auto-generated method
// stub
}
}).setStyle(R.style.CustomShowcaseView)
        .build();
}
@Override
```

```
                                public void
onShowcaseViewDidHide(ShowcaseView showcaseView) {

                                }
                        })

                        .build();

        }

        private void showTutorial2ndStep() {
                new ShowcaseView.Builder(LocationChooserActivity.this)
                        .setTarget(
                                    new
ActionItemTarget(LocationChooserActivity.this,
                                                R.id.spinnerRadius))
                        .setContentTitle("How to get results")
                        .setContentText(
                                    "Then, select a search radius from
the next list e.g 5km.\nFinally,click the search button to get the
results.\nNote: Click on the menu on the top right of the screen to
get more help, change settings or log-out from the application. ")

        .hideOnTouchOutside().setStyle(R.style.CustomShowcaseView)
                        .build();
        }

        private boolean isFirstTime() {
                SharedPreferences preferences =
getPreferences(MODE_PRIVATE);
                boolean ranBefore = preferences.getBoolean("RanBefore",
false);
                if (!ranBefore) {
                        // first time
                        SharedPreferences.Editor editor =
preferences.edit();
                        editor.putBoolean("RanBefore", true);
                        editor.commit();
                }
                return !ranBefore;
```

```
        }

        @Override
        public void onLocationChanged(Location location) {

                double latitude = location.getLatitude();
                double longitude = location.getLongitude();
                LatLng latLng = new LatLng(latitude, longitude);

                map.addMarker(
                                new MarkerOptions()
                                                .position(latLng)
                                                .icon(BitmapDescriptorFactory

        .defaultMarker(BitmapDescriptorFactory.HUE_AZURE)))
                                .setTitle("You are here");
                if (firstTimeLocation) {

        map.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                        map.animateCamera(CameraUpdateFactory.zoomTo(15));
                        firstTimeLocation = false;
                }

        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                getMenuInflater().inflate(R.menu.menu_location_chooser,
menu);
                Spinner s = (Spinner)
menu.findItem(R.id.spinnerType).getActionView();
                ArrayAdapter<String> adapterOptions = new
ArrayAdapter<String>(
                                LocationChooserActivity.this,
                                android.R.layout.simple_spinner_dropdown_item,
options);
                s.setAdapter(adapterOptions); // set the adapter
                s.setOnItemSelectedListener(new OnItemSelectedListener() {
```

```
			@Override
			public void onItemSelected(AdapterView<?> arg0, View
arg1, int pos,
					long arg3) {
				if (pos == 0) {
					option = null;
					return;
				} else {
					option = options[pos];
				}
				if (option.compareToIgnoreCase("Search for
sensor types") == 0) {

	SpinSensorType.setVisibility(View.VISIBLE);
				} else {
					SpinSensorType.setVisibility(View.GONE);
				}

			}

			@Override
			public void onNothingSelected(AdapterView<?> arg0) {

			}

		});
		Spinner s2 = (Spinner) menu.findItem(R.id.spinnerRadius)
				.getActionView();
		ArrayAdapter<String> adapterRadius = new
ArrayAdapter<String>(
				LocationChooserActivity.this,
				android.R.layout.simple_spinner_dropdown_item,
radius_opt);
		s2.setAdapter(adapterRadius); // set the adapter
		s2.setOnItemSelectedListener(new OnItemSelectedListener()
{

			@Override
			public void onItemSelected(AdapterView<?> parent,
View view,
```

```
                              int position, long id) {
                        if (position > 0) {
                              radius =
Integer.parseInt(radius_opt[position]);
                        }
                  }

                  @Override
                  public void onNothingSelected(AdapterView<?> parent)
{
                  }

            });
            if (isFirstTime()) {
                  showTutorial();
            }

            return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == R.id.log_out) {
                  Intent intent = new
Intent(LocationChooserActivity.this,
                              LauncherActivity.class);
                  Utilities.applicationUser = null;

                  intent.setFlags(intent.getFlags() |
Intent.FLAG_ACTIVITY_NO_HISTORY);
                  startActivity(intent);
                  finish();

            }
            if (id == R.id.action_search) {
                  if (option == null || radius == -1 || selected ==
null) {
                        Toast.makeText(getApplicationContext(),
```

```
                                    "Please enter search options and
location",
                                    Toast.LENGTH_LONG).show();
                        return false;
                  }
                  if (option == "Search for controllers") {
                        GetResultsSortByController grsc = new
GetResultsSortByController();
                        try {
                              resultSearch = grsc.execute("").get();
                        } catch (InterruptedException e) {

                              e.printStackTrace();
                        } catch (ExecutionException e) {

                              e.printStackTrace();
                        }

                        Intent intent = new
Intent(LocationChooserActivity.this,
                                    MainActivity.class);

                        intent.putExtra("type", option);
                        intent.putExtra("Result", resultSearch);

                        startActivity(intent);
                  } else if (option == "Search for sensors") {

                        try {
                              resultSearch = new
GetResultsSortBySensor().execute("")
                                          .get();
                        } catch (InterruptedException e) {

                              e.printStackTrace();
                        } catch (ExecutionException e) {

                              e.printStackTrace();
                        }
```

```
                        Intent intent = new
Intent(LocationChooserActivity.this,
                                MainActivity.class);

                        intent.putExtra("type", option);
                        intent.putExtra("Result", resultSearch);
                        startActivity(intent);

                  } else {
                        if (sensorType == null) {
                              Toast.makeText(getApplicationContext(),
                                          "Please enter search options
and location",
                                          Toast.LENGTH_LONG).show();
                              return false;
                        }
                        try {
                              resultSearch = new
GetResultsSortBySensor().execute("")
                                          .get();
                        } catch (InterruptedException e) {

                              e.printStackTrace();
                        } catch (ExecutionException e) {

                              e.printStackTrace();
                        }
                        Intent intent = new
Intent(LocationChooserActivity.this,
                                MainActivity.class);

                        intent.putExtra("type", option);
                        intent.putExtra("sensor_type", sensorType);
                        intent.putExtra("Result", resultSearch);
                        startActivity(intent);

                  }
            }
            if (id == R.id.help) {
```

```
                  AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                              LocationChooserActivity.this);
                  alertDialogBuilder.setTitle("Help");
                  alertDialogBuilder.setCancelable(true);
                  alertDialogBuilder
                              .setMessage("Click on the map to add a
sensor or a controller.\n"
                                          + "Long click on the map to
select the search location");
                  alertDialogBuilder.setNegativeButton("Cancel",
                              new DialogInterface.OnClickListener() {
                                    public void
onClick(DialogInterface dialog, int id) {
                                          dialog.cancel();
                                    }
                              });
                  alertDialogBuilder.show();
            }
            if (id == R.id.action_loc_chooser_settings) {
                  AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                              LocationChooserActivity.this);
                  alertDialogBuilder.setTitle("Change Base Settings");
                  alertDialogBuilder.setCancelable(true);
                  LayoutInflater inflater =
LocationChooserActivity.this
                              .getLayoutInflater();
                  final View dialoglayout = inflater.inflate(
                              R.layout.location_chooser_settings,
null);
                  alertDialogBuilder.setView(dialoglayout);
                  TextView baseurl = (TextView) dialoglayout

      .findViewById(R.id.textViewLocChooserSettingsDefUrl);
                  baseurl.setText("Default Base URL: " +
Utilities.DEFAULT_BASE_URL);
                  TextView currUrl = (TextView) dialoglayout
                              .findViewById(R.id.textViewUsername);
```

```
                    currUrl.setText(currUrl.getText() + " Current Base
URL is: "
                                + Utilities.API_BASE_URL);
                    TextView buid = (TextView) dialoglayout

      .findViewById(R.id.textViewAnyplaceBuildID);
                    buid.setText("Change building ID (Anyplace). Current
ID is: "
                                + Utilities.API_ANYPLACE_BUILDING_ID);
                    TextView floor = (TextView) dialoglayout

      .findViewById(R.id.textViewAnyplaceFloor);
                    floor.setText("Change floor number (Anyplace).
Current floor number is: "
                                + Utilities.API_ANYPLACE_FLOOR);
                    alertDialogBuilder.setNegativeButton("Cancel",
                                new DialogInterface.OnClickListener() {
                                      public void
onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();

                                      }
                                });
                    alertDialogBuilder.setPositiveButton("Save", new
OnClickListener() {

                          @Override
                          public void onClick(DialogInterface dialog,
int which) {

                                EditText newUrl = (EditText)
dialoglayout

      .findViewById(R.id.editTextLocChooserNewUrl);
                                if
(!newUrl.getText().toString().equals("")) {

      Utilities.changeBaseUrl(newUrl.getText().toString());
                                }
```

```
                              Log.i("Response",
Utilities.API_BASE_URL);
                              EditText newbuid = (EditText)
dialoglayout

      .findViewById(R.id.editTextAnyplaceBuildID);
                              if
(!newbuid.getText().toString().equals("")) {
                                    Utilities

      .changeBuildingID(newbuid.getText().toString());
                              }
                              EditText newfloor = (EditText)
dialoglayout

      .findViewById(R.id.editTextAnyplaceFloor);
                              if
(!newfloor.getText().toString().equals("")) {

      Utilities.changeBuilingFloor(Integer.parseInt(newfloor

      .getText().toString()));
                              }
                              Log.i("Response",
Utilities.API_ANYPLACE_BUILDING_ID + " "
                                          +
Utilities.API_ANYPLACE_FLOOR
                                          + Utilities.API_BASE_URL);
                              Intent intent = new
Intent(LocationChooserActivity.this,
                                          LauncherActivity.class);
                              Utilities.applicationUser = null;
                              Toast.makeText(getApplicationContext(),
                                          "Login again for the effects
to take place",
                                          Toast.LENGTH_LONG).show();
                              try {
                                    new LogOut().execute("").get();
                              } catch (InterruptedException e) {
```

```
                                        e.printStackTrace();
                                } catch (ExecutionException e) {

                                        e.printStackTrace();
                                }

                                intent.setFlags(intent.getFlags()
                                                |
Intent.FLAG_ACTIVITY_NO_HISTORY);
                                startActivity(intent);
                                finish();

                        }
                });
                alertDialogBuilder.setNeutralButton("Reset to
Default",
                                new OnClickListener() {

                                        @Override
                                        public void
onClick(DialogInterface dialog, int which) {


        Utilities.changeBaseUrl(Utilities.DEFAULT_BASE_URL);
                                                Utilities

        .changeBuildingID(Utilities.API_DEFAULT_ANYPLACE_BUILDING_ID);
                                                Utilities

        .changeBuilingFloor(Utilities.API_DEFAULT_ANYPLACE_FLOOR);
                                                Intent intent = new Intent(

        LocationChooserActivity.this,

        LauncherActivity.class);
                                                Utilities.applicationUser =
null;

                                                Toast.makeText(

        getApplicationContext(),
```

```
                                                            "Login again for
the effects to take place",

      Toast.LENGTH_LONG).show();
                                          try {
                                                new
LogOut().execute("").get();
                                          } catch
(InterruptedException e) {
                                                e.printStackTrace();
                                          } catch (ExecutionException
e) {

                                                e.printStackTrace();
                                          }


      intent.setFlags(intent.getFlags()
                                                      |
Intent.FLAG_ACTIVITY_NO_HISTORY);
                                          startActivity(intent);
                                          finish();

                                    }
                              });
                  alertDialogBuilder.show();
            }

            return true;
      }

      @Override
      public void onResume() {
            mapView.onResume();

            super.onResume();
      }

      @Override
      public void onDestroy() {
```

```java
        super.onDestroy();
        mapView.onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        mapView.onLowMemory();
    }

    private void assignResult(String res) {
        resultSearch = res;
    }

    private void showProgressDialog() {
        progDailog = new
ProgressDialog(LocationChooserActivity.this);
        progDailog.setMessage("Loading...");
        progDailog.setIndeterminate(false);
        progDailog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        progDailog.setCancelable(true);
        progDailog.show();
    }

    class GetResultsSortByController extends AsyncTask<String, Void,
String> {

        @Override
        protected void onPreExecute() {

        }

        @Override
        protected void onPostExecute(String result) {

            resultSearch = result.toString();
            assignResult(result);
        }

        @Override
```

```
protected String doInBackground(String... arg0) {
        URL url;
        Log.i("Response", "yolo");
        InputStream is = null;
        DataInputStream dis = null;
        String s;
        StringBuilder sb = new StringBuilder();
        String jsonString = null;
        HttpURLConnection urlc = null;

        try {
                url = new
URL(Utilities.API_SORT_BY_CONTROLLER);
                Log.i("Response",
Utilities.API_SORT_BY_CONTROLLER);
                urlc = (HttpURLConnection)
url.openConnection();
                urlc.setRequestMethod("POST");
                urlc.setDoInput(true);
                urlc.setDoOutput(true);
                urlc.setRequestProperty("Content-Type",
                                "application/x-www-form-
urlencoded");
                OutputStream os = urlc.getOutputStream();

                BufferedWriter writer = new BufferedWriter(
                                new OutputStreamWriter(os));
                writer.write("token=" + TOKEN);
                writer.write("&longitude=" +
selected.longitude);
                writer.write("&latitude=" +
selected.latitude);
                writer.write("&radius=" + radius);
                writer.flush();
                writer.close();
                urlc.connect();
                dis = new DataInputStream(new
BufferedInputStream(
                                urlc.getInputStream()));
                while ((s = dis.readLine()) != null) {
```

```
                    sb = sb.append(s + "\n");
                }
            } catch (MalformedURLException mue) {
                mue.printStackTrace();
            } catch (IOException ioe) {
                ioe.printStackTrace();
            } finally {
                if (is != null) {
                    try {
                        is.close();
                    } catch (IOException ioe) {
                        ioe.printStackTrace();
                    }
                }
            }

            jsonString = sb.toString();
            Log.i("Response", jsonString);
            return jsonString;
        }

    }

    class GetResultsSortBySensor extends AsyncTask<String, Void,
String> {

        @Override
        protected void onPreExecute() {
        }

        @Override
        protected void onPostExecute(String result) {
            resultSearch = result.toString();
            assignResult(result);
        }

        @Override
        protected String doInBackground(String... arg0) {
            URL url;
```

```
                    InputStream is = null;
                    DataInputStream dis = null;
                    String s;
                    StringBuilder sb = new StringBuilder();
                    String jsonString = null;
                    HttpURLConnection urlc = null;

                    try {
                          url = new URL(Utilities.API_SORT_BY_SENSOR);

                          urlc = (HttpURLConnection)
url.openConnection();
                          urlc.setRequestMethod("POST");
                          urlc.setDoInput(true);
                          urlc.setDoOutput(true);
                          urlc.setRequestProperty("Content-Type",
                                      "application/x-www-form-
urlencoded");
                          OutputStream os = urlc.getOutputStream();

                          BufferedWriter writer = new BufferedWriter(
                                      new OutputStreamWriter(os));
                          writer.write("token=" + TOKEN);
                          writer.write("&longitude=" +
selected.longitude);
                          writer.write("&latitude=" +
selected.latitude);
                          writer.write("&radius=" + radius);
                          writer.flush();
                          writer.close();
                          urlc.connect();
                          Log.i("Response", TOKEN);
                          Log.i("Response", selected.longitude + "");
                          Log.i("Response", selected.latitude + "");
                          Log.i("Response", radius + "");
                          dis = new DataInputStream(new
BufferedInputStream(
                                      urlc.getInputStream()));
                          while ((s = dis.readLine()) != null) {
                                sb = sb.append(s + "\n");
```

```
                }
            } catch (MalformedURLException mue) {
                mue.printStackTrace();
            } catch (IOException ioe) {
                ioe.printStackTrace();
            } finally {
                if (is != null) {
                    try {
                        is.close();
                    } catch (IOException ioe) {
                        ioe.printStackTrace();
                    }
                }
            }

            jsonString = sb.toString();
            return jsonString;
        }

    }

    class GetPOIsAnyplace extends AsyncTask<String, Void, String> {

        @Override
        protected void onPreExecute() {

            showProgressDialog();

        }

        @Override
        protected String doInBackground(String... arg0) {
            URL url;

            InputStream is = null;
            DataInputStream dis = null;
            String s;
            StringBuilder sb = new StringBuilder();
            String jsonString = null;
            HttpURLConnection urlc = null;
```

```
                  Log.i("Response", "here");
                  try {
                        url = new
URL(Utilities.API_ANYPLACE_GET_POIS);

                        urlc = (HttpURLConnection)
url.openConnection();
                        urlc.setRequestMethod("POST");
                        urlc.setDoInput(true);
                        urlc.setDoOutput(true);
                        urlc.setRequestProperty("Content-Type",
"application/json");
                        OutputStream os = urlc.getOutputStream();

                        BufferedWriter writer = new BufferedWriter(
                                    new OutputStreamWriter(os));
                        writer.write("{\"buid\":\""
                                    +
Utilities.API_ANYPLACE_BUILDING_ID
                                    + "\",\"floor_number\":\""
                                    + Utilities.API_ANYPLACE_FLOOR +
"\"}");
                        writer.flush();
                        writer.close();
                        urlc.connect();
                        Log.i("Response", urlc.getResponseMessage());
                        dis = new DataInputStream(new
BufferedInputStream(
                                    urlc.getInputStream()));
                        while ((s = dis.readLine()) != null) {
                              sb = sb.append(s + "\n");
                        }
                  } catch (MalformedURLException mue) {
                        mue.printStackTrace();
                  } catch (IOException ioe) {
                        ioe.printStackTrace();
                  } finally {
                        if (is != null) {
                              try {
                                    is.close();
```

```java
                        } catch (IOException ioe) {
                            ioe.printStackTrace();
                        }
                    }
                }

                jsonString = sb.toString();
                Log.i("Response", jsonString);
                return jsonString;
            }

            @Override
            protected void onPostExecute(String result) {
                if (progDailog != null && progDailog.isShowing()) {
                    progDailog.dismiss();
                }

                if (result != null) {

                    JSONObject jsonObject = null;
                    JSONArray pois = null;

                    try {

                        jsonObject = new JSONObject(result);
                        String status =
jsonObject.getString("status");
                        if (!status.equals("success")) {

    Toast.makeText(LocationChooserActivity.this,

    jsonObject.getString("Data not retrieved"),

    Toast.LENGTH_LONG).show();
                            return;
                        }
                        pois = jsonObject.getJSONArray("pois");
                        int i = 0;
                        JSONObject poi = null;
```

```
                              while ((poi = pois.getJSONObject(i)) !=
null) {
                                    double latitude =
poi.getDouble("coordinates_lat");
                                    Log.i("Response", latitude +
"latitude");
                                    double longitude =
poi.getDouble("coordinates_lon");
                                    LatLng sensor_poi = new
LatLng(latitude, longitude);
                                    Log.i("Response",
sensor_poi.longitude + " longitude");
                                    map.addMarker(new MarkerOptions()
                                                .position(sensor_poi)

      .title(poi.getString("name") + " Anyplace POI")

      .icon(BitmapDescriptorFactory

      .defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));
                                    i++;
                              }

                        } catch (JSONException je) {
                              return;
                        }
                  }

            }
      }

      class LogOut extends AsyncTask<String, Void, String> {

            @Override
            protected void onPreExecute() {
                  // showProgressDialog();
            }

            @Override
            protected void onPostExecute(String result) {
```

```
		}

		@Override
		protected String doInBackground(String... arg0) {
			URL url;

			InputStream is = null;
			DataInputStream dis = null;
			String s;
			StringBuilder sb = new StringBuilder();
			String jsonString = null;
			HttpURLConnection urlc = null;

			try {
				url = new URL(Utilities.API_LOGOUT);
				urlc = (HttpURLConnection)
url.openConnection();
				urlc.setRequestMethod("POST");
				urlc.setDoInput(true);
				urlc.setDoOutput(true);
				urlc.setRequestProperty("Content-Type",
						"application/x-www-form-
urlencoded");
				OutputStream os = urlc.getOutputStream();

				BufferedWriter writer = new BufferedWriter(
						new OutputStreamWriter(os));
				writer.write("token=" + TOKEN);
				writer.flush();
				writer.close();
				urlc.connect();
				dis = new DataInputStream(new
BufferedInputStream(
						urlc.getInputStream()));
				while ((s = dis.readLine()) != null) {
					sb = sb.append(s + "\n");
				}
			} catch (MalformedURLException mue) {
				mue.printStackTrace();
```

```
			} catch (IOException ioe) {
				ioe.printStackTrace();
			} finally {
				if (is != null) {
					try {
						is.close();
					} catch (IOException ioe) {
						ioe.printStackTrace();
					}
				}
			}

			jsonString = sb.toString();
			if (jsonString == null) {
				return "fail";
			}
			JSONObject jsonObject = null;
			try {
				jsonObject = new JSONObject(jsonString);
			} catch (JSONException e1) {

				e1.printStackTrace();
			}
			try {
				if (jsonObject == null) {
					return "fail";
				}
				if
(!jsonObject.getString("status").equals("success")) {
					return "fail";
				}
			} catch (JSONException e) {

				e.printStackTrace();
			}
			return "success";
		}

	}
```

```java
	class GetSensorTypes extends AsyncTask<String, Void, String[]> {

		@Override
		protected void onPreExecute() {
			// showProgressDialog();

		}

		@Override
		protected void onPostExecute(String[] result) {

		}

		@Override
		protected String[] doInBackground(String... arg0) {
			URL url;
			InputStream is = null;
			DataInputStream dis = null;
			String s;
			StringBuilder sb = new StringBuilder();
			String jsonString = null;
			HttpURLConnection urlc = null;

			try {
				url = new URL(Utilities.API_GET_SENSOR_TYPES);
				urlc = (HttpURLConnection)
url.openConnection();
				urlc.setRequestMethod("POST");
				urlc.setDoInput(true);
				urlc.setDoOutput(true);
				urlc.setRequestProperty("Content-Type",
						"application/x-www-form-
urlencoded");
				OutputStream os = urlc.getOutputStream();

				BufferedWriter writer = new BufferedWriter(
						new OutputStreamWriter(os));
				writer.write("token=" + TOKEN);
				writer.flush();
				writer.close();
```

```
                        urlc.connect();
                        dis = new DataInputStream(new
BufferedInputStream(
                                    urlc.getInputStream()));
                        while ((s = dis.readLine()) != null) {
                              sb = sb.append(s + "\n");
                        }
                  } catch (MalformedURLException mue) {
                        mue.printStackTrace();
                  } catch (IOException ioe) {
                        ioe.printStackTrace();
                  } finally {
                        if (is != null) {
                              try {
                                    is.close();
                              } catch (IOException ioe) {
                                    ioe.printStackTrace();
                              }
                        }
                  }
                  String[] types = null;
                  jsonString = sb.toString();
                  Log.i("Response", jsonString);
                  if (jsonString != null) {

                        try {
                              JSONObject jsonObject = new
JSONObject(jsonString);

                              int size = jsonObject.getInt("count");
                              types = new String[size + 1];
                              types[0] = "Sensor Type";
                              JSONArray s_types =
jsonObject.getJSONArray("result");
                              for (int i = 0; i < size; i++) {
                                    JSONObject type =
s_types.getJSONObject(i);
                                    types[i + 1] =
type.getString("Property");
                              }
```

```
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                } else {
                    return null;
                }
                return types;
            }

        }

        class UploadSensor extends AsyncTask<String, Void, String> {

            @Override
            protected void onPreExecute() {
                // showProgressDialog();

            }

            @Override
            protected void onPostExecute(String result) {

            }

            @Override
            protected String doInBackground(String... arg0) {
                URL url;
                InputStream is = null;
                DataInputStream dis = null;
                String s;
                StringBuilder sb = new StringBuilder();
                String jsonString = null;
                HttpURLConnection urlc = null;
                String typeid = arg0[0];
                String sensorname = arg0[1];

                try {
                    url = new URL(Utilities.API_UPLOAD_SENSOR);
                    urlc = (HttpURLConnection)
url.openConnection();
```

```
                        urlc.setRequestMethod("POST");
                        urlc.setDoInput(true);
                        urlc.setDoOutput(true);
                        urlc.setRequestProperty("Content-Type",
                                    "application/x-www-form-
urlencoded");
                        OutputStream os = urlc.getOutputStream();

                        BufferedWriter writer = new BufferedWriter(
                                    new OutputStreamWriter(os));
                        writer.write("token=" + TOKEN);
                        writer.write("&sensorName=" + sensorname);
                        writer.write("&type=" + typeid);
                        Log.i("Response", typeid);
                        writer.flush();
                        writer.close();
                        urlc.connect();
                        dis = new DataInputStream(new
BufferedInputStream(
                                    urlc.getInputStream()));
                        while ((s = dis.readLine()) != null) {
                              sb = sb.append(s + "\n");
                        }
                  } catch (MalformedURLException mue) {
                        mue.printStackTrace();
                  } catch (IOException ioe) {
                        ioe.printStackTrace();
                  } finally {
                        if (is != null) {
                              try {
                                    is.close();
                              } catch (IOException ioe) {
                                    ioe.printStackTrace();
                              }
                        }
                  }
                  jsonString = sb.toString();
                  Log.i("Response", jsonString);
                  return jsonString;
```

```
            }
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle
extras) {

        }

        @Override
        public void onProviderEnabled(String provider) {

        }

        @Override
        public void onProviderDisabled(String provider) {

        }

}
```

8.2 MainActivity class

```
package com.example.tabbedmonitor;

import java.io.BufferedInputStream;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.LinkedList;
import java.util.concurrent.ExecutionException;
```

```java
import org.achartengine.chart.PointStyle;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import
com.example.tabbedmonitor.LocationChooserActivity.GetResultsSortByCont
roller;
import com.github.amlcurran.showcaseview.OnShowcaseEventListener;
import com.github.amlcurran.showcaseview.ShowcaseView;
import com.github.amlcurran.showcaseview.targets.ActionViewTarget;
import com.github.amlcurran.showcaseview.targets.ViewTarget;
import com.google.android.gms.internal.bu;

import android.app.Activity;
import android.app.ActionBar;
import android.app.AlertDialog;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.DialogInterface.OnClickListener;
import android.content.res.ColorStateList;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Typeface;
import android.support.v13.app.FragmentPagerAdapter;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.view.ViewPager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
```

```java
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.Spinner;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends Activity implements
ActionBar.TabListener {

      /**
       * The {@link android.support.v4.view.PagerAdapter} that will
provide
       * fragments for each of the sections. We use a {@link
FragmentPagerAdapter}
       * derivative, which will keep every loaded fragment in memory.
If this
       * becomes too memory intensive, it may be best to switch to a
       * {@link android.support.v13.app.FragmentStatePagerAdapter}.
       */
      SectionsPagerAdapter mSectionsPagerAdapter;

      /**
       * The {@link ViewPager} that will host the section contents.
       */
      ViewPager mViewPager;
      int currentTab = 0;
      public static LinkedList<SensorLogItem> data_chart;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
```

```
		final ActionBar actionBar = getActionBar();
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		// Set up the action bar.

	actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

		// Create the adapter that will return a fragment for each
of the three
		// primary sections of the activity.
		mSectionsPagerAdapter = new
SectionsPagerAdapter(getFragmentManager());

		// Set up the ViewPager with the sections adapter.
		mViewPager = (ViewPager) findViewById(R.id.pager);
		mViewPager.setOffscreenPageLimit(3);
		mViewPager.setAdapter(mSectionsPagerAdapter);

		// When swiping between different sections, select the
corresponding
		// tab. We can also use ActionBar.Tab#select() to do this
if we have
		// a reference to the Tab.
		mViewPager
				.setOnPageChangeListener(new
ViewPager.SimpleOnPageChangeListener() {
					@Override
					public void onPageSelected(int position)
{

	actionBar.setSelectedNavigationItem(position);
					}
				});

		if (getIntent().getStringExtra("type").compareTo(
				"Search for controllers") == 0) {
			String jsonString =
getIntent().getStringExtra("Result");
```

```
                    if (jsonString.compareTo("") == 0 || jsonString ==
null) {
                          Toast.makeText(getApplicationContext(),
                                      "Unable to retrieve results",
Toast.LENGTH_LONG).show();
                          finish();
                    } else {
                          Log.i("Response", jsonString);
                          parseSearchForControllers(jsonString);
                    }
              } else if
(getIntent().getStringExtra("type").compareToIgnoreCase(
                          "Search for sensors") == 0) {
                    String jsonString =
getIntent().getStringExtra("Result");

                    if (jsonString.compareTo("") == 0 || jsonString ==
null) {
                          Toast.makeText(getApplicationContext(),
                                      "Unable to retrieve results",
Toast.LENGTH_LONG).show();
                          finish();
                    }
                    Log.i("Response", jsonString);
                    parseSearchForSensors(jsonString);
              } else {
                    String jsonString =
getIntent().getStringExtra("Result");
                    if (jsonString.compareTo("") == 0 || jsonString ==
null
                                || 
(getIntent().getStringExtra("sensor_type")) == null) {
                          Toast.makeText(getApplicationContext(),
                                      "Unable to retrieve results",
Toast.LENGTH_LONG).show();
                          finish();
                    }
                    Log.i("Response", jsonString);
                    parseSearchForSensorTypes(jsonString,
```

```
        getIntent().getStringExtra("sensor_type"));
                }
                // For each of the sections in the app, add a tab to the
action bar.
                for (int i = 0; i < mSectionsPagerAdapter.getCount(); i++)
{
                        actionBar.addTab(actionBar.newTab()

        .setText(mSectionsPagerAdapter.getPageTitle(i))
                                        .setTabListener(this));

                }
                getActionBar().setDisplayHomeAsUpEnabled(true);

        }

        private boolean isFirstTime() {
                SharedPreferences preferences =
getPreferences(MODE_PRIVATE);
                boolean ranBefore = preferences.getBoolean("RanBefore",
false);
                if (!ranBefore) {
                        // first time
                        SharedPreferences.Editor editor =
preferences.edit();
                        editor.putBoolean("RanBefore", true);
                        editor.commit();
                }
                return !ranBefore;

        }

        private void showTutorial() {
                ViewTarget target = new
ViewTarget(R.id.listViewArduinoSensors, this);
                new ShowcaseView.Builder(MainActivity.this)
                                .setTarget(target)

                                .setContentTitle("Click on an item")
```

```
                            .setContentText(
                                    "Click on an item from the list of
sensors to deactivate the sensor or view history values as a list or a
chart")

    .hideOnTouchOutside().setStyle(R.style.CustomShowcaseView)
                            .build();

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if
it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        if (isFirstTime()) {
            new ShowcaseView.Builder(this)
                    .setTarget(
                            new ActionViewTarget(this,

    ActionViewTarget.Type.OVERFLOW))
                    .setContentTitle("More menu options")
                    .setContentText(
                            "Click the overflow button
to change settings or get more help")

    .hideOnTouchOutside().setStyle(R.style.CustomShowcaseView)
                    .setShowcaseEventListener(new
OnShowcaseEventListener() {

                        @Override
                        public void
onShowcaseViewShow(ShowcaseView showcaseView) {
                            // TODO Auto-generated
method stub

                        }

                        @Override
```

```
                                        public void
onShowcaseViewHide(ShowcaseView showcaseView) {
                                              // TODO Auto-generated
method stub
                                              new
ShowcaseView.Builder(MainActivity.this)
                                                        .setTarget(
                                                                  new
ActionViewTarget(

            MainActivity.this,

            ActionViewTarget.Type.HOME))

      .setContentTitle("Go back")
                                                        .setContentText(

      "Click the back button to go to the previous screen")

      .hideOnTouchOutside()

      .setStyle(R.style.CustomShowcaseView)

      .setShowcaseEventListener(
                                                                  new
OnShowcaseEventListener() {


      @Override

      public void onShowcaseViewShow(

                  ShowcaseView showcaseView) {

            // TODO Auto-generated

            // method

            // stub
```

```
}

@Override
public void onShowcaseViewHide(
        ShowcaseView showcaseView) {
    // TODO Auto-generated
    // method
    // stub
    showTutorial();
}

@Override
public void onShowcaseViewDidHide(
        ShowcaseView showcaseView) {
    // TODO Auto-generated
    // method
    // stub

}
}).build();
                              }
```

```
                                          @Override
                                          public void onShowcaseViewDidHide(
                                                      ShowcaseView
showcaseView) {
                                                // TODO Auto-generated
method stub

                                          }
                                    }).build();
                  }
                  return true;
            }

            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                  // Handle action bar item clicks here. The action bar will
                  // automatically handle clicks on the Home/Up button, so
long
                  // as you specify a parent activity in
AndroidManifest.xml.
                  int id = item.getItemId();
                  if (id == R.id.action_about) {
                        if
((mSectionsPagerAdapter.fragments.get(currentTab).getClass()) ==
RaspberryPiFragment.class) {
                              AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                                          MainActivity.this);
                              alertDialogBuilder
                                          .setTitle("Info about raspberry"
                                                      +
((RaspberryPiFragment) mSectionsPagerAdapter.fragments

            .get(currentTab)).id);
                              alertDialogBuilder.setCancelable(true);
                              RaspberryPiFragment lf = (RaspberryPiFragment)
mSectionsPagerAdapter.fragments
                                          .get(currentTab);
                              LinkedList<SensorLogItem> data = lf.data;
```

```
                        String message = "Currently logged sensors
are: \n";
                        for (int i = 0; i < data.size(); i++) {
                              message += data.get(i).sensor + " " +
"on "
                                          + data.get(i).source + "\n";
                        }
                        alertDialogBuilder.setMessage(message);
                        alertDialogBuilder.setNegativeButton("Cancel",
                                    new
DialogInterface.OnClickListener() {
                                          public void
onClick(DialogInterface dialog, int id) {
                                                dialog.cancel();
                                          }
                                    });
                        alertDialogBuilder.show();

                  }
                  if
((mSectionsPagerAdapter.fragments.get(currentTab).getClass()) ==
ArduinoFragment.class) {
                        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                                    MainActivity.this);
                        alertDialogBuilder.setTitle("Info about
arduino "
                                    + ((ArduinoFragment)
mSectionsPagerAdapter.fragments
                                                .get(currentTab)).id);
                        alertDialogBuilder.setCancelable(true);
                        ArduinoFragment lf = (ArduinoFragment)
mSectionsPagerAdapter.fragments
                                    .get(currentTab);
                        LinkedList<SensorLogItem> data = lf.data;
                        String message = "Currently logged sensors
are: \n";
                        for (int i = 0; i < data.size(); i++) {
                              message += data.get(i).sensor + " " +
"on "
```

```
                                        + data.get(i).source + "\n";
                        }
                        alertDialogBuilder.setMessage(message);
                        alertDialogBuilder.setNegativeButton("Cancel",
                                    new
DialogInterface.OnClickListener() {
                                          public void
onClick(DialogInterface dialog, int id) {
                                                dialog.cancel();
                                          }
                                    });
                        alertDialogBuilder.show();

                  }
                  if
((mSectionsPagerAdapter.fragments.get(currentTab).getClass()) ==
LogsFragment.class) {
                        LogsFragment lf = (LogsFragment)
mSectionsPagerAdapter.fragments
                                    .get(currentTab);
                        LinkedList<SensorLogItem> data = lf.data;
                        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                                    MainActivity.this);
                        alertDialogBuilder.setTitle("Info about
sensors ");
                        alertDialogBuilder.setCancelable(true);
                        String message = "Currently logged sensors
are: \n";
                        for (int i = 0; i < data.size(); i++) {
                              message += data.get(i).sensor + " " +
"on "
                                          + data.get(i).source + "\n";
                        }
                        alertDialogBuilder.setMessage(message);
                        alertDialogBuilder.setNegativeButton("Cancel",
                                    new
DialogInterface.OnClickListener() {
                                          public void
onClick(DialogInterface dialog, int id) {
```

```
                                                dialog.cancel();
                                          }
                                   });
                       alertDialogBuilder.show();
                 }
           }
           if (id == R.id.action_refresh) {
                 Intent intent = getIntent();
                 String result = null;

                 if (getIntent().getStringExtra("type").compareTo(
                             "Search for controllers") == 0) {
                       LocationChooserActivity lc = new
LocationChooserActivity();

      LocationChooserActivity.GetResultsSortByController c = lc.new
GetResultsSortByController();
                       try {
                             result = c.execute("").get();
                       } catch (InterruptedException e) {

                             e.printStackTrace();
                       } catch (ExecutionException e) {

                             e.printStackTrace();
                       }
                 } else {
                       LocationChooserActivity lc = new
LocationChooserActivity();
                       LocationChooserActivity.GetResultsSortBySensor
c = lc.new GetResultsSortBySensor();

                       try {
                             result = c.execute("").get();
                       } catch (InterruptedException e) {

                             e.printStackTrace();
                       } catch (ExecutionException e) {

                             e.printStackTrace();
```

```
                        }
                }
                intent.putExtra("Result", result);
                intent.setFlags(intent.getFlags() |
Intent.FLAG_ACTIVITY_NO_HISTORY);
                finish();

                startActivity(intent);
                return true;
            }

            if (id == android.R.id.home) {
                Intent homeIntent = new Intent(this,
LocationChooserActivity.class);
                homeIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(homeIntent);
                finish();
            }
            if (id == R.id.action_help) {
                AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
                        MainActivity.this);
                alertDialogBuilder.setTitle("Help");
                alertDialogBuilder.setCancelable(true);
                String message = "Click on a sensor to
deactivate/view history\n"
                        + "Click on \"View History\" and then
\"Show Chart\" to see the"
                        + " values represented as a chart";

                alertDialogBuilder.setMessage(message);
                alertDialogBuilder.setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void
onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });
                alertDialogBuilder.show();
```

```
        }

        if (id == R.id.action_settings) {
            if (Utilities.applicationUser.getUserType() ==
UserTypes.simple) {
                Toast.makeText(getApplicationContext(),
                        "You do not have permission to
change settings",
                        Toast.LENGTH_LONG).show();
                return false;
            }
            if
(mSectionsPagerAdapter.fragments.get(currentTab).getClass() !=
LogsFragment.class) {
                LayoutInflater inflater = getLayoutInflater();
                View dialoglayout =
inflater.inflate(R.layout.settings_screen,
                        null);
                SeekBar sb = (SeekBar)
dialoglayout.findViewById(R.id.seekBar1);
                Switch sw = (Switch)
dialoglayout.findViewById(R.id.switch1);
                sw.setChecked(true);
                sb.setOnSeekBarChangeListener(new
OnSeekBarChangeListener() {
                    int progressChanged = 0;

                    public void onProgressChanged(SeekBar
seekBar,
                            int progress, boolean
fromUser) {
                        progressChanged = progress;
                    }

                    public void onStartTrackingTouch(SeekBar
seekBar) {
                    }

                    public void onStopTrackingTouch(SeekBar
seekBar) {
```

```
                                    Toast.makeText(MainActivity.this,
                                            "seek bar progress:" +
progressChanged,

      Toast.LENGTH_SHORT).show();
                              }
                        });
                        AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                        builder.setView(dialoglayout);
                        builder.setCancelable(true);
                        builder.setTitle("Settings for controller ");
                        builder.setPositiveButton("Save", new
OnClickListener() {

                              @Override
                              public void onClick(DialogInterface
dialog, int which) {
                              }
                        });
                        builder.setNegativeButton("Cancel", new
OnClickListener() {

                              @Override
                              public void onClick(DialogInterface
dialog, int which) {
                                    dialog.dismiss();
                              }
                        });
                        builder.show();
                  } else {
                        LayoutInflater vi = (LayoutInflater)
getApplicationContext()

      .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                        final View v = vi

      .inflate(R.layout.sensor_settings_screen, null);

                        // fill in any details dynamically here
```

```
                        LinearLayout linear = (LinearLayout) v

      .findViewById(R.id.settings_layout);
                        AlertDialog.Builder builder = new
AlertDialog.Builder(this);

                        LogsFragment lf = (LogsFragment)
mSectionsPagerAdapter
                                    .getItem(currentTab);
                        final LinkedList<SensorLogItem> data =
lf.data;
                        for (int i = 0; i < data.size(); i++) {
                              TextView tv = new
TextView(getApplicationContext());
                              tv.setText(data.get(i).sensor + " on " +
data.get(i).source);
                              tv.setTextSize(15);
                              tv.setTextColor(Color.BLACK);
                              linear.addView(tv);
                              SeekBar sb = new
SeekBar(getApplicationContext());
                              sb.setMax(20);
                              sb.setId(i);
                              sb.setProgress(data.get(i).frequency);
                              sb.setOnSeekBarChangeListener(new
OnSeekBarChangeListener() {
                                    int progressChanged = 0;

                                    public void
onProgressChanged(SeekBar seekBar,
                                                int progress, boolean
fromUser) {
                                          progressChanged = progress;
                                    }

                                    public void
onStartTrackingTouch(SeekBar seekBar) {
                                    }
```

```
                                    public void
onStopTrackingTouch(SeekBar seekBar) {

      Toast.makeText(MainActivity.this,
                                                      "seek bar
progress:" + progressChanged,

      Toast.LENGTH_SHORT).show();
                                    }
                              });
                              linear.addView(sb);

                        }
                        builder.setTitle("Set sensor frequency");
                        builder.setView(v);
                        builder.setPositiveButton("Save", new
OnClickListener() {

                              @Override
                              public void onClick(DialogInterface
dialog, int which) {
                                    for (int i = 0; i < data.size();
i++) {

                                          SeekBar sb = (SeekBar)
v.findViewById(i);

                                          Log.i("Response",
sb.getProgress() + "");
                                          SetSensorFrequency ssf = new
SetSensorFrequency();
                                          ssf.execute(data.get(i).id +
"", sb.getProgress()
                                                      + "");

                                    }

                              }
                        });
```

```
                        builder.setNegativeButton("Cancel", new
OnClickListener() {

                              @Override
                              public void onClick(DialogInterface
dialog, int which) {

                                    dialog.dismiss();
                              }
                        });
                        builder.show();
                  }
            }
            return super.onOptionsItemSelected(item);
      }

      @Override
      public void onTabSelected(ActionBar.Tab tab,
                  FragmentTransaction fragmentTransaction) {
            // When the given tab is selected, switch to the
corresponding page in
            // the ViewPager.
            mViewPager.setCurrentItem(tab.getPosition());
            currentTab = tab.getPosition();
            getActionBar().setTitle(

      mSectionsPagerAdapter.getPageTitle(tab.getPosition()));
      }

      @Override
      public void onTabUnselected(ActionBar.Tab tab,
                  FragmentTransaction fragmentTransaction) {
      }

      @Override
      public void onTabReselected(ActionBar.Tab tab,
                  FragmentTransaction fragmentTransaction) {
      }

      /**
```

```
	 * A {@link FragmentPagerAdapter} that returns a fragment
corresponding to
	 * one of the sections/tabs/pages.
	 */
	public class SectionsPagerAdapter extends FragmentPagerAdapter {

		LinkedList<Fragment> fragments = new
LinkedList<Fragment>();

		public SectionsPagerAdapter(FragmentManager fm) {
			super(fm);

		}

		@Override
		public Fragment getItem(int position) {
			return fragments.get(position);

		}

		@Override
		public int getCount() {
			return fragments.size();
		}

		@Override
		public CharSequence getPageTitle(int position) {

			if (fragments.get(position).getClass() ==
RaspberryPiFragment.class) {
				return ("Raspberry" + " " +
((RaspberryPiFragment) fragments
						.get(position)).id);
			}
			if (fragments.get(position).getClass() ==
LogsFragment.class) {
				return ("Sensors List");
			}
			if (fragments.get(position).getClass() ==
ArduinoFragment.class) {
```

```
                        return ("Arduino " + ((ArduinoFragment)
fragments.get(position)).id);
                  }
                  return null;
            }
      }

      public static class ArduinoFragment extends Fragment {
            int id;
            static int total = 0;
            public String ip;
            public String time;
            public String location;
            ListView sensors_list;
            LinkedList<SensorLogItem> data;

            public ArduinoFragment() {
                  id = 1;
                  data = new LinkedList<>();
            }

            @Override
            public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                        Bundle savedInstanceState) {
                  View v = inflater.inflate(R.layout.fragment_arduino,
container,
                              false);
                  Typeface font =
Typeface.createFromAsset(getActivity().getAssets(),
                              "fontawesome-webfont.ttf");

                  TextView button = (TextView) v
                              .findViewById(R.id.textViewArdMapIcon);
                  button.setTypeface(font);
                  sensors_list = (ListView) v

      .findViewById(R.id.listViewArduinoSensors);
                  TextView iptv = (TextView)
v.findViewById(R.id.textViewRaspIp);
```

```
                    iptv.setText(ip);
                    TextView timetv = (TextView)
v.findViewById(R.id.textViewArdTime);
                    timetv.setText(time);
                    timetv.setTextSize(15);
                    TextView loctv = (TextView) v
                            .findViewById(R.id.textViewArdLocation);
                    loctv.setText(location);
                    loctv.setTextSize(15);
                    LogsListAdapter la = new
LogsListAdapter(getActivity(), data);
                    sensors_list.setAdapter(la);
                    sensors_list.setOnItemClickListener(new
OnItemClickListener() {

                        @Override
                        public void onItemClick(AdapterView<?> parent,
View view,
                                int position, long id) {
                            final SensorLogItem li = (SensorLogItem)
parent

      .getItemAtPosition(position);
                            AlertDialog.Builder alertDialogBuilder =
new AlertDialog.Builder(
                                    getActivity());
                            alertDialogBuilder.setTitle("Info about
sensor "
                                    + li.sensor);
                            alertDialogBuilder.setCancelable(true);
                            alertDialogBuilder
                                    .setMessage("This sensor is
currently active.\nChoose what to do with this sensor");

      alertDialogBuilder.setNegativeButton("Cancel",
                                    new
DialogInterface.OnClickListener() {
                                        public void
onClick(DialogInterface dialog,
                                                int id) {
```

```
                                                        dialog.cancel();
                                                  }
                                             });

      alertDialogBuilder.setNeutralButton("View History",
                                             new OnClickListener() {

                                                  @Override
                                                  public void
onClick(DialogInterface dialog,
                                                             int which)
{

                                                        MainActivity ma
= new MainActivity();
                                                        data_chart =
ma.getHistory(li.id + "");

                                                        LayoutInflater
inflater = getActivity()

      .getLayoutInflater();
                                                        View
dialoglayout = inflater.inflate(

      R.layout.fragment_logs, null);
                                                        ListView lv =
(ListView) dialoglayout

      .findViewById(R.id.listViewArduinoSensors);
                                                        LogsListAdapter
la = new LogsListAdapter(

      getActivity(), data_chart);

      lv.setAdapter(la);

      AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
```

```
        getActivity());

        alertDialogBuilder

        .setTitle("History about sensor "

              + li.sensor);

        alertDialogBuilder.setView(dialoglayout);


        alertDialogBuilder.setCancelable(true);

        alertDialogBuilder.setNegativeButton(

        "Close", new OnClickListener() {


        @Override

        public void onClick(

                    DialogInterface dialog,

                    int which) {


              dialog.cancel();

        }
                                                                   });

        alertDialogBuilder.setPositiveButton(

        "Show chart",
                                                                    new
  OnClickListener() {
```

```
      @Override
      public void onClick(
                  DialogInterface dialog,
                  int which) {
            Intent intent = new Intent(
                        getActivity(),
                        ShowChartActivity.class);
            intent.putExtra(
                        "Sensor type",
                        li.sensor);
            startActivity(intent);

      }
                                                            });

      alertDialogBuilder.show();

                                          }
                                    });

      alertDialogBuilder.setPositiveButton("Deactivate",
                                    new OnClickListener() {

                                          @Override
                                          public void
onClick(DialogInterface dialog,
                                                      int which)
{
```

```
                                                    if
(Utilities.applicationUser.getUserType() == UserTypes.simple) {

     Toast.makeText(

     getActivity()

                  .getApplicationContext(),

     "You do not have permission to change settings",

     Toast.LENGTH_LONG).show();

     dialog.cancel();
                                                         return;
                                                    }
                                                    MainActivity ma
= new MainActivity();
                                                    DeactivateSensor
ds = ma.new DeactivateSensor();
                                                    ds.execute(li.id
+ "");
                                                    Toast.makeText(

     getActivity()

            .getApplicationContext(),

     li.sensor + " deactivated",

     Toast.LENGTH_LONG).show();

                                                    // Call the api
to deactivate sensor
                                                    dialog.cancel();
                                               }
                                          });
                              alertDialogBuilder.show();

                         }
```

```
                });
                return v;
            }
        }

        public static class RaspberryPiFragment extends Fragment {
            LinkedList<SensorLogItem> data;
            ListView lv;
            public String ip;
            public String time;
            static int total = 0;
            int id;
            public String location;

            public RaspberryPiFragment() {
                super();
                id = 1;

                data = new LinkedList<SensorLogItem>();

            }

            @Override
            public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                        Bundle savedInstanceState) {
                // Inflate the layout for this fragment

                View v =
inflater.inflate(R.layout.fragment_raspberry, container,
                            false);
                Typeface font =
Typeface.createFromAsset(getActivity().getAssets(),
                            "fontawesome-webfont.ttf");

                TextView button = (TextView)
v.findViewById(R.id.textViewMapIcon);
                TextView iptv = (TextView)
v.findViewById(R.id.textViewRaspIp);
                iptv.setText(ip);
```

```
                    TextView timetv = (TextView)
v.findViewById(R.id.textViewRaspTime);
                    timetv.setText(time);
                    timetv.setTextSize(15);
                    TextView loctv = (TextView) v

      .findViewById(R.id.textViewRaspLocation);
                    loctv.setText(location);
                    loctv.setTextSize(15);
                    button.setTypeface(font);

                    lv = (ListView)
v.findViewById(R.id.listViewRaspberrySensors);

                    LogsListAdapter la = new
LogsListAdapter(getActivity(), data);
                    lv.setAdapter(la);
                    lv.setOnItemClickListener(new OnItemClickListener()
{

                          @Override
                          public void onItemClick(AdapterView<?> parent,
View view,
                                      int position, long id) {
                                final SensorLogItem li = (SensorLogItem)
parent

      .getItemAtPosition(position);
                                AlertDialog.Builder alertDialogBuilder =
new AlertDialog.Builder(
                                            getActivity());
                                alertDialogBuilder.setTitle("Info about
sensor "
                                            + li.sensor);
                                alertDialogBuilder.setCancelable(true);
                                alertDialogBuilder
                                            .setMessage("This sensor is
currently active.\nChoose what to do with this sensor");

      alertDialogBuilder.setNeutralButton("View History",
```

```
new OnClickListener() {

@Override
public void
onClick(DialogInterface dialog,
int which)
{

MainActivity ma
= new MainActivity();
data_chart =
ma.getHistory(li.id + "");

LayoutInflater
inflater = getActivity()

      .getLayoutInflater();
View
dialoglayout = inflater.inflate(

      R.layout.fragment_logs, null);
ListView lv =
(ListView) dialoglayout

      .findViewById(R.id.listViewArduinoSensors);
LogsListAdapter
la = new LogsListAdapter(

      getActivity(), data_chart);

      lv.setAdapter(la);

      AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(

      getActivity());

      alertDialogBuilder

      .setTitle("History about sensor "
```

```
		+ li.sensor);
	alertDialogBuilder.setView(dialoglayout);

	alertDialogBuilder.setCancelable(true);
	alertDialogBuilder.setNegativeButton(
	"Close", new OnClickListener() {

	@Override
	public void onClick(
			DialogInterface dialog,
			int which) {

		dialog.cancel();
	}
									});
	alertDialogBuilder.setPositiveButton(
	"Show chart",
									new
OnClickListener() {
	//
	@Override
	public void onClick(
			DialogInterface dialog,
```

```
                int which) {
            Intent intent = new Intent(
                    getActivity(),
                    ShowChartActivity.class);
            intent.putExtra(
                    "Sensor type",
                    li.sensor);
            startActivity(intent);

        }
                                                    });

        alertDialogBuilder.show();

                                        }
                                    });

        alertDialogBuilder.setNegativeButton("Cancel",
                                    new
    DialogInterface.OnClickListener() {
                                        public void
    onClick(DialogInterface dialog,
                                                int id) {
                                            dialog.cancel();
                                        }
                                    });

        alertDialogBuilder.setPositiveButton("Deactivate",
                                    new OnClickListener() {

                                        @Override
```

```
                                                  public void
onClick(DialogInterface dialog,
                                                              int which)
{
                                                        if
(Utilities.applicationUser.getUserType() == UserTypes.simple) {

      Toast.makeText(

      getActivity()

                  .getApplicationContext(),

      "You do not have permission to change settings",

      Toast.LENGTH_LONG).show();

      dialog.cancel();
                                                              return;

                                                        }
                                                        MainActivity ma
= new MainActivity();
                                                        DeactivateSensor
ds = ma.new DeactivateSensor();
                                                        ds.execute(li.id
+ "");
                                                        Toast.makeText(

      getActivity()

            .getApplicationContext(),

      li.sensor + " deactivated",

      Toast.LENGTH_LONG).show();
                                                        dialog.cancel();
                                                  }
                                            });
                                    alertDialogBuilder.show();
```

```
                    }
                });
                return v;
            }

            @Override
            public void onActivityCreated(Bundle savedInstanceState) {
                super.onActivityCreated(savedInstanceState);
            }

    }

    public static class LogsFragment extends Fragment {
            ListView sensors_log;
            LinkedList<SensorLogItem> data;

            public LogsFragment() {
                data = new LinkedList<SensorLogItem>();
            }

            @Override
            public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                        Bundle savedInstanceState) {
                // Inflate the layout for this fragment
                View v = inflater.inflate(R.layout.fragment_logs,
container, false);
                sensors_log = (ListView) v

    .findViewById(R.id.listViewArduinoSensors);

                LogsListAdapter la = new
LogsListAdapter(getActivity(), data);
                sensors_log.setAdapter(la);
                sensors_log.setOnItemClickListener(new
OnItemClickListener() {

                        @Override
```

```
                        public void onItemClick(AdapterView<?> parent,
View view,
                                    int position, long id) {
                              final SensorLogItem li = (SensorLogItem)
parent

      .getItemAtPosition(position);
                              AlertDialog.Builder alertDialogBuilder =
new AlertDialog.Builder(
                                          getActivity());
                              alertDialogBuilder.setTitle("Info about
sensor "
                                          + li.sensor);
                              alertDialogBuilder.setCancelable(true);
                              alertDialogBuilder
                                          .setMessage("This sensor is
currently active.\nChoose what to do with this sensor");

      alertDialogBuilder.setNeutralButton("View History",
                                          new OnClickListener() {

                                                @Override
                                                public void
onClick(DialogInterface dialog,
                                                            int which)
{

                                                      MainActivity ma
= new MainActivity();
                                                      data_chart =
ma.getHistory(li.id + "");

                                                      LayoutInflater
inflater = getActivity()

      .getLayoutInflater();
                                                      View
dialoglayout = inflater.inflate(

      R.layout.fragment_logs, null);
```

```
                                                    ListView lv =
(ListView) dialoglayout

      .findViewById(R.id.listViewArduinoSensors);
                                                    LogsListAdapter
la = new LogsListAdapter(

      getActivity(), data_chart);

      lv.setAdapter(la);

      AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(

      getActivity());

      alertDialogBuilder

      .setTitle("History about sensor "

            + li.sensor);

      alertDialogBuilder.setView(dialoglayout);


      alertDialogBuilder.setCancelable(true);

      alertDialogBuilder.setNegativeButton(

      "Close", new OnClickListener() {


      @Override

      public void onClick(

                  DialogInterface dialog,

                  int which) {
```

```
        dialog.cancel();
    }
                                                            });

    alertDialogBuilder.setPositiveButton(
    "Show chart",
                                                            new
OnClickListener() {

    @Override
    public void onClick(
            DialogInterface dialog,
            int which) {
        Intent intent = new Intent(
                getActivity(),
                ShowChartActivity.class);
        intent.putExtra(
                "Sensor type",
                li.sensor);
        startActivity(intent);

    }
                                                            });

    alertDialogBuilder.show();
```

```
                                    }
                                });

    alertDialogBuilder.setNegativeButton("Cancel",
                                new
DialogInterface.OnClickListener() {
                                    public void
onClick(DialogInterface dialog,
                                            int id) {
                                        dialog.cancel();
                                    }
                                });

    alertDialogBuilder.setPositiveButton("Deactivate",
                                new OnClickListener() {

                                    @Override
                                    public void
onClick(DialogInterface dialog,
                                            int which)
{
                                        if
(Utilities.applicationUser.getUserType() == UserTypes.simple) {

    Toast.makeText(

    getActivity()

            .getApplicationContext(),

    "You do not have permission to change settings",

    Toast.LENGTH_LONG).show();

    dialog.cancel();
                                            return;

                                        }
```

```
                                                            MainActivity ma
= new MainActivity();
                                                            DeactivateSensor
ds = ma.new DeactivateSensor();
                                                            ds.execute(li.id
+ "");
                                                            Toast.makeText(

      getActivity()

            .getApplicationContext(),

      li.sensor + " deactivated",

      Toast.LENGTH_LONG).show();
                                                            dialog.cancel();
                                                      }
                                                });
                                    alertDialogBuilder.show();

                              }
                        });
                        return v;

                  }
            }

            private void parseSearchForControllers(String jsonString) {
                  if (jsonString != null) {
                        if (jsonString != null) {
                              JSONObject jsonObject = null;
                              JSONArray controllers = null;
                              JSONObject sensor = null;
                              JSONArray result = null;

                              try {
                                    jsonObject = new JSONObject(jsonString);
                                    String status =
jsonObject.getString("status");
```

```
                              String time =
jsonObject.getString("request time");
                              String ip = jsonObject.getString("IP
Address");
                              if (!status.equals("success")) {
                                    Toast.makeText(MainActivity.this,

      jsonObject.getString("Data not retrieved"),

      Toast.LENGTH_LONG).show();
                                    return;
                              }
                              String message;
                              message=
jsonObject.getString("message");
                              if (message.equals("None controller was
found within the distance given")){

      Toast.makeText(getApplicationContext(), "No results found at
this location", Toast.LENGTH_LONG).show();
                                    finish();
                              }
                              JSONObject loc = jsonObject
                                          .getJSONObject("Location
Requested");
                              String location = "   " +
loc.getString("latitude") + " , "
                                          +
loc.getString("longitude");
                              int count = jsonObject.getInt("count");
                              controllers =
jsonObject.getJSONArray("controllers");
                              for (int i = 0; i < count; i++) {
                                    JSONObject c =
controllers.getJSONObject(i);
                                    String name =
c.getString("Controller Name");
                                    if (name.contains("arduino")) {
                                          ArduinoFragment a1 = new
ArduinoFragment();
```

```
a1.time = time;
a1.ip = ip;
a1.location = location;
JSONObject res =
c.getJSONObject("Sensors");
int countS =
res.getInt("count");

result =
res.getJSONArray("resultSensors");
for (int j = 0; j < countS;
j++) {
	String start = null;
	String end = null;
	String value = null;
	int frequency = 0;
	int id = 0;
	String timestamp =
null;
	sensor =
result.getJSONObject(j);
	String type =
sensor.getString("Sensor Type");
	id =
sensor.getInt("Sensor ID");
	frequency =
sensor.getInt("SensorFrequency");
	Log.i("Response", type
+ " ");
	if
(type.compareToIgnoreCase("motion") == 0) {
		start =
sensor.getString("Timestamp Start");
		end =
sensor.getString("Timestamp End");
	} else {
		timestamp =

sensor.getString("Timestamp");
		value =
sensor.getString("Measurement");
```

```
        Log.i("Response", value);
                                                    }

                                                    if
(type.compareToIgnoreCase("motion") == 0) {
                                                        a1.data.add(new
SensorLogItem(id, name,

        start, type, end, frequency));

        Log.i("Response", start);
                                                    } else {
                                                        a1.data.add(new
SensorLogItem(id, name,

        timestamp, type, value, frequency));
                                                    }

                                                }

        mSectionsPagerAdapter.fragments.add(a1);

        mSectionsPagerAdapter.notifyDataSetChanged();

                                            } else {
                                                RaspberryPiFragment r1 = new
RaspberryPiFragment();
                                                r1.time = time;
                                                r1.ip = ip;
                                                r1.location = location;
                                                JSONObject res =
c.getJSONObject("Sensors");
                                                int countS =
res.getInt("count");

                                                result =
res.getJSONArray("resultSensors");
                                                for (int j = 0; j < countS;
j++) {
```

```
String start = null;
String end = null;
String value = null;
int id = 0;
String timestamp =
null;
sensor =
result.getJSONObject(j);
String type =
sensor.getString("Sensor Type");
id =
sensor.getInt("Sensor ID");
int frequency = sensor
        .getInt("SensorFrequency");
Log.i("Response", type
+ " ");
if
(type.compareToIgnoreCase("motion") == 0) {
        start =
sensor.getString("Timestamp Start");
        end =
sensor.getString("Timestamp End");
} else {
        timestamp =
sensor.getString("Timestamp");
        value =
sensor.getString("Measurement");
        Log.i("Response", value);
}
if
(type.compareToIgnoreCase("motion") == 0) {
        r1.data.add(new
SensorLogItem(id, name,
        start, type, end, frequency));
        Log.i("Response", start);
```

```
                                                } else {
                                                      r1.data.add(new
SensorLogItem(id, name,

      timestamp, type, value, frequency));
                                                }

                                          }

      mSectionsPagerAdapter.fragments.add(r1);

      mSectionsPagerAdapter.notifyDataSetChanged();
                                    }
                              }

                        } catch (JSONException je) {
                              return;
                        }

                  }
            }
      }

      private void parseSearchForSensors(String jsonString) {
            if (jsonString != null) {

                  JSONObject jsonObject = null;
                  JSONObject sensor = null;
                  JSONArray result = null;

                  try {
                        jsonObject = new JSONObject(jsonString);
                        String status =
jsonObject.getString("status");

                        if (!status.equals("success")) {
                              Toast.makeText(MainActivity.this,
                                          jsonObject.getString("Data
not retrieved"),
                                          Toast.LENGTH_LONG).show();
```

```
                        return;
                    }
                    String message =
jsonObject.getString("message");
                    if (message.equals("None sensor was found
within the distance given")){
                        Toast.makeText(getApplicationContext(),
"No results found at this location", Toast.LENGTH_LONG).show();
                        finish();
                    }
                    int count = jsonObject.getInt("count");
                    Log.i("Response", count + " ");
                    LogsFragment lf = new LogsFragment();
                    result =
jsonObject.getJSONArray("resultSensors");
                    for (int i = 0; i < count; i++) {
                        String start = null;
                        String end = null;
                        String value = null;
                        int id = 0;
                        String timestamp = null;
                        sensor = result.getJSONObject(i);
                        id = sensor.getInt("Sensor ID");
                        int frequency =
sensor.getInt("SensorFrequency");
                        String type = sensor.getString("Sensor
Type");
                        Log.i("Response", type + " ");
                        if (type.compareToIgnoreCase("motion")
== 0) {
                            start =
sensor.getString("Timestamp Start");
                            end = sensor.getString("Timestamp
End");
                        } else {
                            timestamp =
sensor.getString("Timestamp");
                            value =
sensor.getString("Measurement");
```

```
				Log.i("Response", value);
			}

			String microcontroller = sensor
					.getString("Micro-
controllerName");
			if (type.compareToIgnoreCase("motion")
== 0) {
				lf.data.add(new SensorLogItem(id,
microcontroller,
						start, type, end,
frequency));
				Log.i("Response", start);
			} else {
				lf.data.add(new SensorLogItem(id,
microcontroller,
						timestamp, type,
value, frequency));
			}

		}
		if (lf.data.isEmpty()){


	Toast.makeText(getApplicationContext(), "No results found at
this location", Toast.LENGTH_LONG).show();
				finish();
		}
		mSectionsPagerAdapter.fragments.add(lf);
		mSectionsPagerAdapter.notifyDataSetChanged();
	} catch (JSONException je) {
		return;
	}
    }

}

private void parseSearchForSensorTypes(String jsonString, String
sType) {
	if (jsonString != null) {
```

```
JSONObject jsonObject = null;
JSONObject sensor = null;
JSONArray result = null;

try {
    jsonObject = new JSONObject(jsonString);
    String status =
jsonObject.getString("status");

    if (!status.equals("success")) {
        Toast.makeText(MainActivity.this,
                jsonObject.getString("Data
not retrieved"),
                Toast.LENGTH_LONG).show();
        return;
    }
    String message =
jsonObject.getString("message");
    if (message.equals("None sensor was found
within the distance given")){
        Toast.makeText(getApplicationContext(),
"No results found at this location", Toast.LENGTH_LONG).show();
        finish();
    }
    int count = jsonObject.getInt("count");
    Log.i("Response", count + " ");
    LogsFragment lf = new LogsFragment();
    result =
jsonObject.getJSONArray("resultSensors");
    for (int i = 0; i < count; i++) {
        String start = null;
        String end = null;
        String value = null;
        int id = 0;
        String timestamp = null;
        sensor = result.getJSONObject(i);
        id = sensor.getInt("Sensor ID");
        int frequency =
sensor.getInt("SensorFrequency");
```

```
                                        String type = sensor.getString("Sensor
Type");
                                        Log.i("Response", type + " ");
                                        if (type.compareToIgnoreCase(sType) !=
0) {
                                                continue;
                                        }
                                        if (type.compareToIgnoreCase("motion")
== 0) {
                                                start =
sensor.getString("Timestamp Start");
                                                end = sensor.getString("Timestamp
End");
                                        } else {
                                                timestamp =
sensor.getString("Timestamp");
                                                value =
sensor.getString("Measurement");

                                                Log.i("Response", value);
                                        }

                                        String microcontroller = sensor
                                                        .getString("Micro-
controllerName");
                                        if (type.compareToIgnoreCase("motion")
== 0) {
                                                lf.data.add(new SensorLogItem(id,
microcontroller,
                                                                start, type, end,
frequency));
                                                Log.i("Response", start);
                                        } else {
                                                lf.data.add(new SensorLogItem(id,
microcontroller,
                                                                timestamp, type,
value, frequency));
                                        }

                                }
```

```
                        if (lf.data.isEmpty()){

                              Toast.makeText(getApplicationContext(),
"No results found at this location", Toast.LENGTH_LONG).show();
                              finish();
                  }
                        mSectionsPagerAdapter.fragments.add(lf);
                        mSectionsPagerAdapter.notifyDataSetChanged();
                  } catch (JSONException je) {
                        return;
                  }
            }

      }

      public LinkedList<SensorLogItem> getHistory(String sensor) {
            GetHistory gh = new GetHistory();
            try {
                  return gh.execute(sensor).get();
            } catch (InterruptedException e) {

                  e.printStackTrace();
            } catch (ExecutionException e) {

                  e.printStackTrace();
            }
            return null;
      }

      class GetHistory extends AsyncTask<String, Void,
LinkedList<SensorLogItem>> {

            @Override
            protected void onPreExecute() {

            }

            @Override
            protected void onPostExecute(LinkedList<SensorLogItem>
result) {
```

```
            }

            @Override
            protected LinkedList<SensorLogItem>
doInBackground(String... params) {
                  LinkedList<SensorLogItem> list = new
LinkedList<SensorLogItem>();
                  Log.i("Response", params[0]);
                  URL url;
                  Log.i("Response", params[0]);
                  InputStream is = null;
                  DataInputStream dis = null;
                  String s;
                  StringBuilder sb = new StringBuilder();
                  String jsonString = null;
                  HttpURLConnection urlc = null;

                  try {
                        url = new URL(Utilities.API_HISTORY_SENSOR);

                        urlc = (HttpURLConnection)
url.openConnection();
                        urlc.setDoInput(true);
                        urlc.setDoOutput(true);
                        urlc.setRequestMethod("POST");
                        urlc.setDoInput(true);
                        urlc.setDoOutput(true);
                        urlc.setRequestProperty("Content-Type",
                                    "application/x-www-form-
urlencoded");
                        OutputStream os = urlc.getOutputStream();

                        BufferedWriter writer = new BufferedWriter(
                                    new OutputStreamWriter(os));
                        writer.write("token=" +
LocationChooserActivity.TOKEN);
                        writer.write("&sensorID=" + params[0]);

                        writer.flush();
```

```
                        writer.close();
                        urlc.connect();
                        urlc.getResponseCode();

                        dis = new DataInputStream(new
BufferedInputStream(
                                    urlc.getInputStream()));
                        while ((s = dis.readLine()) != null) {
                              sb = sb.append(s + "\n");
                        }
                  } catch (MalformedURLException mue) {
                        mue.printStackTrace();
                  } catch (IOException ioe) {
                        ioe.printStackTrace();
                  } finally {
                        if (is != null) {
                              try {
                                    is.close();
                              } catch (IOException ioe) {
                                    ioe.printStackTrace();
                              }
                        }
                  }

                  jsonString = sb.toString();
                  Log.i("Response", jsonString);
                  if (jsonString != null) {
                        JSONObject jsonObject = null;
                        JSONArray history = null;
                        JSONObject metric = null;

                        try {
                              jsonObject = new JSONObject(jsonString);
                              String status =
jsonObject.getString("status");
                              if (!status.equals("success")) {
                                    Toast.makeText(MainActivity.this,

   jsonObject.getString("Data not retrieved"),
```

```
		Toast.LENGTH_LONG).show();
					return null;
				}
				int count = jsonObject.getInt("count");
				history =
jsonObject.getJSONArray("history Logs");
				for (int i = 0; i < count; i++) {
					JSONObject c =
history.getJSONObject(i);

					String start = null;
					String end = null;
					String value = null;
					String id = null;
					String timestamp = null;

					id = c.getString("Measurement
ID");

					int s_id = c.getInt("Sensor ID");

					String type = c.getString("Sensor
Type");
					if
(type.compareToIgnoreCase("motion") == 0) {
						start =
c.getString("Timestamp Start");
						end = c.getString("Timestamp
End");
						list.add(new
SensorLogItem(s_id, id, start, type,
								end, 0));

					} else {
						value =
c.getString("Measurement");
						timestamp =
c.getString("Timestamp");
						Log.i("Response", value);
```

```
                                          list.add(new
SensorLogItem(s_id, id, timestamp,
                                                      type, value,
0));
                                    }

                              }

                        }

                        catch (JSONException je) {
                              return null;
                        }

                  }

                  return list;
            }

      }

      class SetSensorFrequency extends AsyncTask<String, Void, String>
{

            @Override
            protected void onPreExecute() {
                  // showProgressDialog();

            }

            @Override
            protected void onPostExecute(String result) {

            }

            @Override
            protected String doInBackground(String... arg0) {
                  URL url;
                  InputStream is = null;
                  DataInputStream dis = null;
```

```
			String s;
			StringBuilder sb = new StringBuilder();
			String jsonString = null;
			String sensorID = arg0[0];
			String frequency = arg0[1];
			HttpURLConnection urlc = null;

			try {
				url = new URL(Utilities.API_SET_FREQUENCY);
				urlc = (HttpURLConnection)
url.openConnection();
				urlc.setRequestMethod("POST");
				urlc.setDoInput(true);
				urlc.setDoOutput(true);
				urlc.setRequestProperty("Content-Type",
						"application/x-www-form-
urlencoded");
				OutputStream os = urlc.getOutputStream();

				BufferedWriter writer = new BufferedWriter(
						new OutputStreamWriter(os));
				writer.write("token=" +
Utilities.applicationUser.getToken());
				writer.write("&sensorID=" + sensorID);
				writer.write("&frequency=" + frequency);

				writer.flush();
				writer.close();
				urlc.connect();
				dis = new DataInputStream(new
BufferedInputStream(
						urlc.getInputStream()));
				while ((s = dis.readLine()) != null) {
					sb = sb.append(s + "\n");
				}
			} catch (MalformedURLException mue) {
				mue.printStackTrace();
			} catch (IOException ioe) {
				ioe.printStackTrace();
			} finally {
```

```java
                        if (is != null) {
                            try {
                                is.close();
                            } catch (IOException ioe) {
                                ioe.printStackTrace();
                            }
                        }
                    }

                    jsonString = sb.toString();
                    Log.i("Response", jsonString);
                    return jsonString;
                }

            }

            class DeactivateSensor extends AsyncTask<String, Void, String> {

                @Override
                protected void onPreExecute() {
                    // showProgressDialog();

                }

                @Override
                protected void onPostExecute(String result) {

                }

                @Override
                protected String doInBackground(String... arg0) {
                    URL url;
                    InputStream is = null;
                    DataInputStream dis = null;
                    String s;
                    StringBuilder sb = new StringBuilder();
                    String jsonString = null;
                    String sensorID = arg0[0];

                    HttpURLConnection urlc = null;
```

```
                    try {
                          url = new
URL(Utilities.API_DEACTIVATE_SENSOR);
                          urlc = (HttpURLConnection)
url.openConnection();
                          urlc.setRequestMethod("POST");
                          urlc.setDoInput(true);
                          urlc.setDoOutput(true);
                          urlc.setRequestProperty("Content-Type",
                                      "application/x-www-form-
urlencoded");
                          OutputStream os = urlc.getOutputStream();

                          BufferedWriter writer = new BufferedWriter(
                                      new OutputStreamWriter(os));
                          writer.write("token=" +
Utilities.applicationUser.getToken());
                          writer.write("&sensorID=" + sensorID);

                          writer.flush();
                          writer.close();
                          urlc.connect();
                          dis = new DataInputStream(new
BufferedInputStream(
                                      urlc.getInputStream()));
                          while ((s = dis.readLine()) != null) {
                                sb = sb.append(s + "\n");
                          }
                    } catch (MalformedURLException mue) {
                          mue.printStackTrace();
                    } catch (IOException ioe) {
                          ioe.printStackTrace();
                    } finally {
                          if (is != null) {
                                try {
                                      is.close();
                                } catch (IOException ioe) {
                                      ioe.printStackTrace();
                                }
```

```
                    }
                }

                jsonString = sb.toString();
                Log.i("Response", jsonString);
                return jsonString;
            }

        }
    }
```

9 Παράρτημα B – Κώδικας για συλλογή δεδομένων από αισθητήρες

9.1 Photocell reading

```
//Include arduPi library
#include "arduPi.h"

int photocellPin = 1; // the cell and 10K pulldown are connected to a0
int photocellReading; // the analog reading from the sensor divider
int LEDpin = 11; // connect Red LED to pin 11 (PWM pin)
int LEDbrightness;

void setup()
{
  Wire.begin(); // join i2C bus (address optional for master)
}

void loop()
{
  // channel 0
  photocellReading = analogRead(photocellPin);

printf("Analog reading = %d\n",photocellReading);


// We'll have a few threshholds, qualitatively determined
if (photocellReading < 10) {
Serial.println(" - Dark");
} else if (photocellReading < 200) {
Serial.println(" - Dim");
} else if (photocellReading < 500) {
Serial.println(" - Light");
} else if (photocellReading < 800) {
Serial.println(" - Bright");
} else {
```

```
Serial.println(" - Very bright");
}
delay(1000);
}


int main (){
    setup();
    while(1){
        loop();
    }
    return (0);
}
```

9.2 NTC reading

```
#include "arduPi.h"
#include <math.h>





#define TENKRESISTOR 10000
#define BETA 3977
#define THERMISTOR 2800
#define ROOMTEMPK 298.15
#define READINGS 10


int main (){
    setup ();
    while(1){
        loop();
    }
    return(0);
}
```

```
void setup (){
    printf ("Starting up the thermometer \n");
    Wire.begin();
}

void loop(){
    float avResistance;
    float resistance;
    int combinedreadings[READINGS];
    float kelvin;
    float celsius;
    float fahrenheit;
    float analogReadingArduino;


    /******************
ADC mappings
Pin Address
0 0xDC
1 0x9C
2 0xCC
3 0x8C
4 0xAC
5 0xEC
6 0xBC
7 0xFC
******************/
// 0xDC is our analog 0 pin

/* Grab the two bytes returned from the
analog 0 pin, combine them
and write the value to the
combinedReadings array
*/
for(int r=0; r<READINGS; r++){
    analogReadingArduino=analogRead(0);
combinedreadings[r] = analogReadingArduino;
delay(10); }

// Grab the average of our 7 readings
```

```
// in order to get a more accurate value
avResistance = 0;
for (int r=0; r<READINGS; r++) {
avResistance += combinedreadings[r];
}
avResistance /= READINGS;

/* We can now calculate the resistance of
the readings that have come back from analog 0
*/
avResistance = (1023 / avResistance) - 1;
avResistance = TENKRESISTOR / avResistance;
resistance = avResistance / THERMISTOR;

// Calculate the temperature in Kelvin
kelvin = log(resistance);
kelvin /= BETA;
kelvin += 1.0 / ROOMTEMPK;
kelvin = 1.0 / kelvin;
printf("Temperature in K ");
printf("%f \n",kelvin);

// Convert from Kelvin to Celsius
celsius = kelvin -= 273.15;
printf("Temperature in C ");
printf("%f \n",celsius);

// Convert from Celsius to Fahrenheit
fahrenheit = (celsius * 1.8) + 32;
printf("Temperature in F ");
printf("%f \n",fahrenheit);

// Three second delay before taking our next
// reading
delay(3000);

}
```

10 Παράρτημα Γ – Ερωτήσεις Ερωτηματολογίου Αξιολόγησης

SensoMan Evaluation

I would really appreciate it if you took some time to test my application and answer some questions. The application is based on managing sensor data gathered from our controllers (namely our Raspberry Pi and our Arduino) and is part of my undergraduate thesis.

Please follow the instructions provided to successfully install and start using the application. Some requirements are critical to the proper function of the application so please do not leave steps behind.

Step 1 - Install the app from the link below
https://www.dropbox.com/s/3igip21jag92bts/SensoManage.apk?dl=0

Step 2 - Turn on location services on your phone

Step 3 - Ensure that you are connected to the CS network

Step 4 - Login:
The application requires a log-in for you to proceed. We created three accounts with different properties. The credentials are provided below.

User 1 - Simple User
username: root
password: simple

User 2 - Sensor Owner
username: android-application
password: android

Note:
Some questions require you to do specific steps in the application. Feel free to explore inside the application but please make sure to go back and do these steps when asked.

In the questions, we will be referring to the log-in screen, map screen, main screen and chart screen. These are displayed below so you know which is which

* Required

Εικόνα 31: Ερωτηματολόγιο Αξιολόγησης 1

Αρχικά, στο ερωτηματολόγιο δίνονται κάποιες οδηγίες για εγκατάσταση της εφαρμογής και δίνονται κάποια στοιχεία για σύνδεση στην εφαρμογή για σκοπούς ελέγχου.

1. **Try to search for controllers. How easy was it for you to get any results?**
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

2. **Try to search for sensors. How easy was it for you to get any results?**
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

3. **Try to change some settings in the map screen. How easy was it for you to understand what settings can you change and actually change them?**
(Remember to reset settings to default for the application to work properly)
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

4. **As an admin or sensor owner try to change a sensor's frequency. How easy was it for you to do that?**
You need to log-out and log-in again as an admin or sensor owner
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

5. **As an admin or sensor owner try to change a controller's frequency. How easy was it for you to do that?**
You need to log-out and log-in again as an admin or sensor owner
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

6. **Try to view a sensor's history values. How easy was it for you to to that?**
Mark only one oval.

	1	2	3	4	5	
Very hard	◯	◯	◯	◯	◯	Very easy

Εικόνα 32: Ερωτηματολόγιο Αξιολόγησης 2

7. **Try to view a sensor's history values displayed as a chart. Were you able to read and understand the chart?**
Mark only one oval.

- ( ) Yes
- ( ) No

8. **Try to deactivate a sensor. How easy was it for you to do that?**
(Please do not actually deactivate the sensor)
Mark only one oval.

	1	2	3	4	5	
Very hard	( )	( )	( )	( )	( )	Very easy

9. **Try to add a new controller or sensor. How easy was it for you to do that?**
Mark only one oval.

	1	2	3	4	5	
Very hard	( )	( )	( )	( )	( )	Very easy

10. **Did you find the completion of any of the tasks above particularly difficult? If so please state the task and the reasons that made it difficult for you to complete it.**

____
____
____
____
____

Part 2 - General questions

Please take some time to answer some questions about your general user experience with the application

11. **Did you understand the concept of the application? ***
This means what the application helps you accomplish
Mark only one oval.

- ( ) Yes
- ( ) No

12. **Do you think that this application is a good way to manage your sensors if you ever want to? ***
Mark only one oval.

- ( ) Yes
- ( ) No

Εικόνα 33: Ερωτηματολόγιο Αξιολόγησης 3

13. **Did you know where to find help if you needed to?**
Mark only one oval.

- Yes
- No

14. **How easy was it for you to navigate between different screens?**
Mark only one oval.

	1	2	3	4	5	
Very hard						Very easy

15. **How helpful were the on-screen instructions and/or help messages?**
Mark only one oval.

	1	2	3	4	5	
Not helpful at all						Very helpful

16. **Did you use the "About" button at all?**
Mark only one oval.

- Yes
- No

17. **On the map screen could you find any Anyplace POIs?**
(POIs = points of interest)
Mark only one oval.

- Yes
- No

18. **Leave any suggestions/concerns about the functionality of the application**
Anything you would like to see included, anything you would like to be implemented differently

Εικόνα 34: Ερωτηματολόγιο Αξιολόγησης 4

19. **Did the application crash at any point? If so please state the steps you took and the screen you were on.**

20. **In your opinion, how useful could this application be to you?** *

Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Very useful

21. **Can you think of a situation that this application could be applied to?** *

e.g Manage the sensors of an automated security system

22. **Final thoughts.**

Please state anything that you would like me to know or your general idea of the application

Εικόνα 35: Ερωτηματολόγιο Αξιολόγησης 5

Όπως φαίνεται και από τις εικόνες πιο πάνω το ερωτηματολόγιο αποτελείται από 22 ερωτήσεις που έχουν να κάνουν με τη λειτουργία της εφαρμογής αλλά και με την ευχρηστία και χρησιμότητα της εφαρμογής. Μερικές από τις ερωτήσεις είναι ξεκάθαρα για περεταίρω έλεγχο της εφαρμογής για σφάλματα ενώ κάποιες προορίζονται για αξιολόγηση της αποδοχής της εφαρμογής από τους χρήστες.